SPACE ELEVATOR • BIT BANGING • BOT CONSTRUCTION SETS

SERVO MAGAZINE

FOR THE ROBOT EXPERIMENTER

ROBOT Soccer Champion

• HACK A LASER POINTER

• USING INTERRUPTS TO CONTROL SERVOS

• ROBOSAPIEN V2 REVIEW

February 2006

www.servomagazine.com

ROBOTICS HiTEC

Assimilate This...

Hitec Robotics introduces ROBONOVA-I

The remarkable ROBONOVA-1 is an exciting humanoid robot offering educators, students and robotic hobbyists a complete and ultra modern robot package.

The kit contains a detailed English manual and all components necessary to build, program and operate your own robot.

- Remocon IR Controller
- Sixteen HSR-8498HB servos
- MR-C3024 Micro controller board
- Re-chargeable NiMH battery and charger
- Robo Basic programming software and manual
- PC Serial port programming interface cable

MR-C3024 - Robot Controller (#78001)

Using the latest mega 128 MPU controller from ATMEL, the stable ROBONOVA-1 can walk, do flips, cartwheels, dance moves, and is ready to compete in any Robo One class "J" competition.

HSR-8498HB - Hitec Digital Robot Servo (#38498)

This fully articulating, 12" high, mechanical man is controlled with 16 powerful Hitec HSR-8498HB digital servos built specifically for the ROBONOVA-1. These custom servos feature super strong Karbonite gear trains and "feedback" technology for easy programming.

ROBONOVA-I

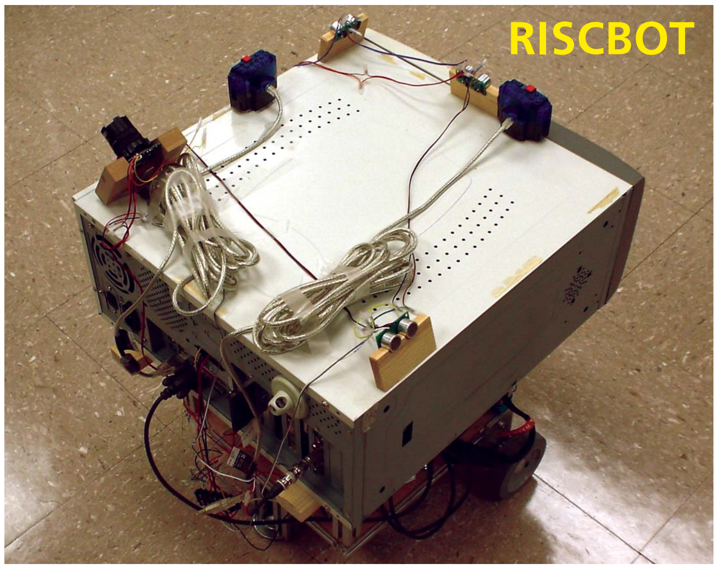
Available as a kit **[#77000]**
or pre-assembled, RTW [Ready To Walk] package **[#77002]**

For more information on availability and pricing, call us at 858-748-6948 or visit,
www.hitecrobotics.com

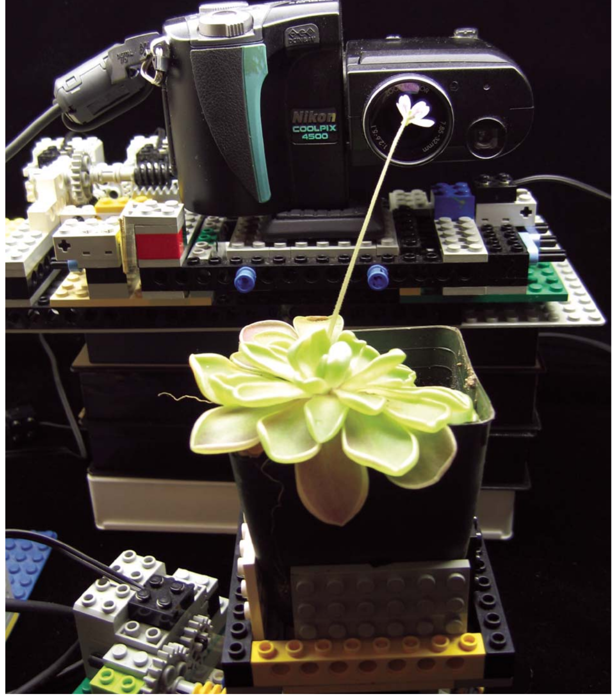
SERVO

Features & Projects

Coming 03.2006



Jasper the 3D Movie Bot



SERVO Magazine **(ISSN 1546-0592/CDN Pub Agree#40702530)** is published monthly for $24.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. APPLICATION TO MAIL AT PERIODICALS POSTAGE RATE IS PENDING AT CORONA, CA AND AT ADDITIONAL ENTRY MAILING OFFICES. POSTMASTER: Send address changes to **SERVO Magazine, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@servomagazine.com

02.2006
VOL. 4 NO. 2

Departments

Columns

ON THE COVER

Mind / Iron

by Michael Simpson

I have been building robots of some sort or another for as long as I can remember. I have purchased nearly every robotics book that has been printed. There is an unbelievable amount of satisfaction in seeing the robot you have been working on for the last few months come to life and navigate around an obstacle or wall, even if it happens to go through the wall. Many of you have seen the on-board camera video of my Duct Bot do a face dive into the plenum of my furnace. Very cool.

We have come a long way in the last 20 years or so. For one, we have the Internet. The Internet has now made it easier than ever to research any aspect of the hobby. I can't count the number of times I have been able to cut countless hours off my research by a few simple searches on the web. You will even discover complete projects on the Internet, and they can be found on both private and commercial websites. The Internet also allows us to purchase many of the products we would not otherwise have access to. In my day, this was called mail-order and all transactions were done via snail mail.

There is one other aspect of the Internet that you won't find in any book. This is called *community*. A community is not just an online forum as there are hundreds of those. It's when you have a group of individuals who frequently help one another and are willing to help others. In an online community, we welcome others to join and share their own ideas. We are all too familiar with the negative aspects of the Internet such as spam and viruses. When it comes to hobbies, I have found that online communities allow you to experience your hobby like you never could before. Sure you can join a robotics club; but many of us live in areas where these just don't exist.

Let's take a look at a possible scenario: You're browsing through a magazine or surfing the web and you see a great project you would like to build. It looks interesting and might be the perfect project for you and your son (or daughter) to build together. One problem: you have not built a project of this magnitude and feel it might be a bit overwhelming.

Now suppose the magazine or website has an online forum that encourages discussions on its various projects or articles. You take a look at a few of the forums and see many levels of discussion ranging from beginner to advanced. Great!

You check out that project to see what you need. Oh no! Another problem! You notice that one of the key components is a motor assembly from a 67 Mustang windshield wiper. There is no way you are going to find one of these. After looking through one of the forums about this project you notice that others are having the same problem. Wow. You've found something. Someone else has built the same project with some of those new VEX motors that are available at just about any local RadioShack. He even posts a website with pictures. Armed with this additional information and very available parts, you decide this is a project you could easily build. It's time to go and get the parts.

While you were at RadioShack, you noticed that all sorts of robot

Mind/Iron Continued →

Published Monthly By
T & L Publications, Inc.
430 Princeland Court
Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Product Order Line **1-800-783-4624**
www.servomagazine.com

Subscriptions
Inside US **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277
North Hollywood, CA 91615

PUBLISHER
Larry Lemieux
publisher@servomagazine.com

ASSOCIATE PUBLISHER/ VP OF SALES/MARKETING
Robin Lemieux
display@servomagazine.com

CIRCULATION DIRECTOR
Tracy Kerley
subscribe@servomagazine.com

WEB CONTENT/STORE
Michael Kaudze
sales@servomagazine.com

PRODUCTION/GRAPHICS
Shannon Lemieux
Michele Durant

BIO-FEEDBACK

Dear SERVO:

I am afraid the translation of the sentence "Défense de fumer" is not quite adequate (even if I don't imagine this could lead to confusion when reading the article!).

Rather than "it is smoking," I think a more appropriate translation would be "It is forbidden to smoke" (misunderstood by the servo driver!).

Keep producing an interesting magazine!

Jea-Marc Pacouret, France

Dear SERVO:

You publish a great magazine. I can't wait to receive it every month. However, your editors and James Antonakos should brush up their French skills. At least they should use Google's language tools. The subtitle to James' article – Défense de fumer – means "Smoking forbidden" and not "It is smoking." I'm sure James can otherwise tell a joke.

Cheers!

Jean-Yves Allard, Montreal QC

accessories are available. Things like light sensors, servos, wheels, and even complete kits. You pick up a couple motors and head home. You initially complete your project but you decide to head back to the Shack and pick up a couple of the VEX sonar range sensors. After some experimentation (and some community help), you get your enhanced project complete. You (and your son or daughter) are so impressed, you decide to place the results of your project up on the forum for the benefit of others and a whole new line of discussions begin. Congratulations! You have just completed the robotics circle of life.

These are very exciting times in the robotics field. New things are happening practically every day. And online communities, readily-available robot components, inexpensive entry level microcontrollers, and computer interfaces are making it easy to get involved in this leading edge hobby.

If you haven't built any robot projects recently, pick up a magazine or browse the web. You will be pleasantly surprised at what is available. We are living in a very exciting time right now. Now, all you have to do is get off that couch, shut off the TV, and get started. **SV**

ROBOTICS RESOURCES

Tune in each month for a heads-up on where to get all of your "robotics resources" for the best prices!

BY GORDON MCCOMB

Exploring Robotics Construction Sets

Construction toys have always been big sellers. Who hasn't at least played around with parts from an Erector set, LEGO, Capsela, or other construction kit, snapping or screwing together various pieces, and experimenting with their own creations? These toys have proven to be perennial favorites because they allow individual creative freedom; you're not locked into someone else's idea of what a car, motorcycle, flying saucer, or robot looks like.

Amateur robotics has long relied on bits and pieces from these construction kits as a "raidable" store of small, lightweight, and relatively inexpensive parts. The resulting contraptions sometimes bear a resemblance to the worst of Rube Goldberg inventions, but they're nevertheless workable and affordable. Not every robot builder enjoys a budget of thousands or even hundreds of dollars for custom-machined parts.

Adding to the mix of store-bought construction kits is a small but growing cadre of specialty building components expressly designed for small robotics applications. Several companies are now offering bits and pieces to build desktop robots, where these bits and pieces are custom made to interface to the components we use the most — R/C servos, small wheels, sensors, and more.

While the idea of universal construction parts isn't new, the new line of robot-centric components is a welcome addition to those of us who like to "roll our own." These parts allow us to build our own custom robot, but without expensive or difficult custom machining. In this column, we'll look at some of these robotics construction kits, and while we're at it, review the old standbys — LEGO, Erector Set, and the others — that are still available.

VEX Robotics Design System

Aimed at both the educational and hobbyist market, the VEX system is based around the Erector set style of pre-drilled girders and connector pieces. Most parts are fastened using traditional techniques of machine screws and nuts. What sets VEX apart from a traditional Erector or other construction toy set is that it contains pieces specially designed for small robotics. It comes with two types of motors made to fit the girder construction of the system (servo and continuous rotation); to these motors you can attach a variety of mechanical parts, including gears, idlers, wheels, and tank treat drive sprockets.

VEX is sold — online and through RadioShack, among other sources — as a complete construction system, including a custom microcontroller, radio controlled transmitter, and a variety of mechanical sensor switches. You can use the VEX sets as-is, or incorporate the parts in your own designs. You can purchase a starter kit, or choose from a variety of accessories and additional parts: a sprocket and tread set for converting your robot to a tank design, wheels, and a novel sprocket and adjustable-length chain set.

Lynxmotion Servo Erector Set

Most desktop robots use radio control servo motors of one type or another. These motors are fairly inexpensive, and can be used out-of-the-box as servos, or reconfigured to turn continuously. Operating a servo is fairly simple, and requires just an R/C transmitter and receiver, a microcontroller, or even a timer circuit based around the 555 IC.

Considering the popularity of R/C servos in robotics, Lynxmotion's Servo Erector Set is an idea long overdue. It's nice to see Lynxmotion address this market. The Servo Erector Set is composed of various brackets and other hardware for the express purpose of connecting together standard-size R/C servo motors. You can connect these brackets to a traditional robot base to build a rolling or walking machine, or attach them to tubes, hubs, and connectors to fashion completely free-form designs. The Lynxmotion website provides a number of examples of prototypes constructed from their line of Servo Erector Set parts, including hexapod walkers and bipedal walkers, arms, and even wheeled self-balancing bots.

80/20 Extruded Aluminum

Billed as the "Industrial Erector Set," 80/20's line of extruded aluminum provides a convenient — if not somewhat pricey — method of assembling larger robots with a minimum amount

of custom design work. Extruded aluminum is composed of bars (and other shapes) of aluminum; the extrusion process creates small grooves in the aluminum to which you can attach various connectors and other construction pieces. You merely cut the aluminum bars to length, then fasten them together with the available connector pieces.

80/20, Inc., is not the only company that offers extruded aluminum, but they are among the most popular. You can find the stuff at many industrial parts outlets, such as Reid Tool & Supply, or even locally.

Ye Olde Standbys

Here's a quick rundown of the more popular toy construction sets, all of which make for a rich source of parts.

Erector

Erector has been around for almost a century. The kits were made of all-metal, but now contain a number of plastic pieces. The sets come in various sizes, and are generally designed to build a number of different projects. Many kits are engineered for a specific design with perhaps, provisions for moderate variations. Useful components of the kits include pre-punched metal girders, plastic and metal plates, tires, wheels, shafts, and plastic mounting panels. You can use any as you see fit, assembling your robots with the hardware supplied with the kit, or with 6-32 or 8-32 nuts and bolts.

Several Erector sets come with wheels, construction beams, and other assorted parts that you can use to construct a robot base. Motors are typically not included in these kits, but you can readily supply your own. Because Erector packages regularly come and go, what follows is a general guide to building a robot base. You'll need to adapt and reconfigure based on the Erector parts you have on hand.

Over the years, the Erector brand has gone through many owners. Parts from old Erector sets are unlikely to fit well with new parts, including but not limited to differences in the threads used for the nuts and bolts. Similarly, today's Meccano sets are only passably compatible with the English-made Meccano sets sold decades ago. Hole spacing and sizes have varied over the years, and "mixing and matching" is not practical, or desirable.

LEGO

LEGO has become the premier construction toy, for both children and adults. Apart from the ever-popular Robotics Invention System set — which is expressly designed to build a robot — you can use LEGO pieces to construct whole robots, or parts of robots. The parts snap together, but for more permanent creations, you can use a dollop of ABS solvent cement.

MEGA BLOKS

The MEGA BLOKS toys use a similar design to LEGO, and the construction pieces are more-or-less "LEGO compatible." One use of MEGA BLOKS is as a low-cost alternative for some basic LEGO pieces, but for the robot builder, you'll be interested in some of their specialty products that come along every once in a while; these are often highly suited to the purpose of amateur robotics. For example, their now discontinued Battle Bloks kits used a six-wheel "all terrain" design, along with dual motors. You can make your MEGA BLOKS constructions more permanent with a tab of modeler's styrene solvent cement.

Capsela

Capsela is a popular snap-together motorized parts kit that uses unusual tube and sphere shapes. Capsela kits come in different sizes and have one or more gear motors that can be attached to various components. The kits contain unique parts that other put-together toys don't, such as plastic chain and chain sprockets/gears. Advanced kits come with remote control and computer circuits. All the parts from the various kits are interchangeable.

Fischertechnik

The Fischertechnik kits are in Germany and imported into North America by a small number of companies. "Toy" isn't the proper term for them, because the Fischertechnik kits are not designed for use by small children. In fact, many of the kits are meant for junior high school through college

FIGURE 1. Aluminum extrusions from 80/20, Inc., may be used to construct large, sturdy robots.

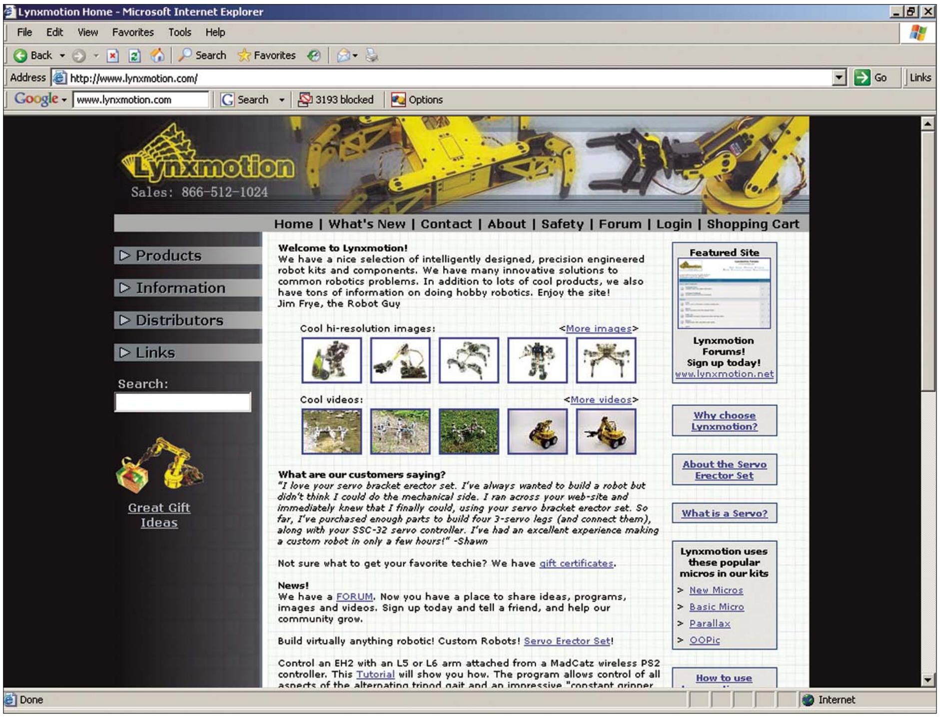


FIGURE 2. Lynxmotion's line of Servo Erector Set brackets and other parts allow you to make servo-based contraptions of your own design.

industrial engineering students, and offer a snap-together approach to making working electromagnetic, hydraulic, pneumatic, static, and robotic mechanisms. Because of the cost of the Fischertechnik kits, you may not want to cannibalize them for robot components. But if you are interested in learning more about mechanical theory and design, the Fischertechnik kits used as-is provide a thorough and programmed method for jumping in with both feet.

FIGURE 3. Hobby Engineering offers a rich assortment of Erector set kits.

K'Nex

K'Nex uses unusual half-round plastic spokes and connector rods to build things of all descriptions. You can construct a robot with just K'Nex parts, or use the parts in a larger, mixed-component robot. The base of a walking robot may be made from a thin sheet of aluminum, but the legs might be constructed from various K'Nex pieces, for example.

A number of K'Nex kits are available, from simple starter sets to rather massive special-purpose collections (many of which are designed to build robots, dinosaurs, or robot-dinosaurs). Several of the kits come with small gear motors so you can motorize your creation. The motors are also available separately.

Sources for Robot Construction Sets

80/20, Inc.
www.8020inc.net

Aluminum extrusions and connection parts for industrial-strength constructions. Useful for larger robots. Check the site for local retailers.

Amazon.com
www.amazon.com

Best-known as a book seller, Amazon also sells toys through affiliations with Toys R Us and Imaginarium. The latter specializes in unique educational products.

Construction Toys
www.constructiontoys.com

Online and local retailer of construction toys. These toys are available both online and in retail stores: Capsela; Eitech; Erector; Fischertechnik; Geofix; Geomag; K'NEX; LEGO Dacta; Roger's Connection; Rhomblocks; Rokenbok; Zome System.

e-Hobbyland
e-hobbyland.com

Well-established retail and online seller of all types of toys.

Hobby Engineering
www.hobbyengineering.com

General source for robot parts, as well as a lengthy list of Erector set kits of all shapes and sizes.

KBtoys.com
www.kbtoys.com *OR*
www.etoys.com

Online mail order. Check often for deep discounts on LEGO, K'NEX, and other brands.

LEGO Shop-at-Home
shop.lego.com

Online outlet for LEGO products, including spare parts (when available).

Lynxmotion
www.lynxmotion.com

Lynxmotion offers complete robot kits, as well as a unique Servo Erector Set: a collection of brackets and other parts for building custom robots using standard-size R/C servos.

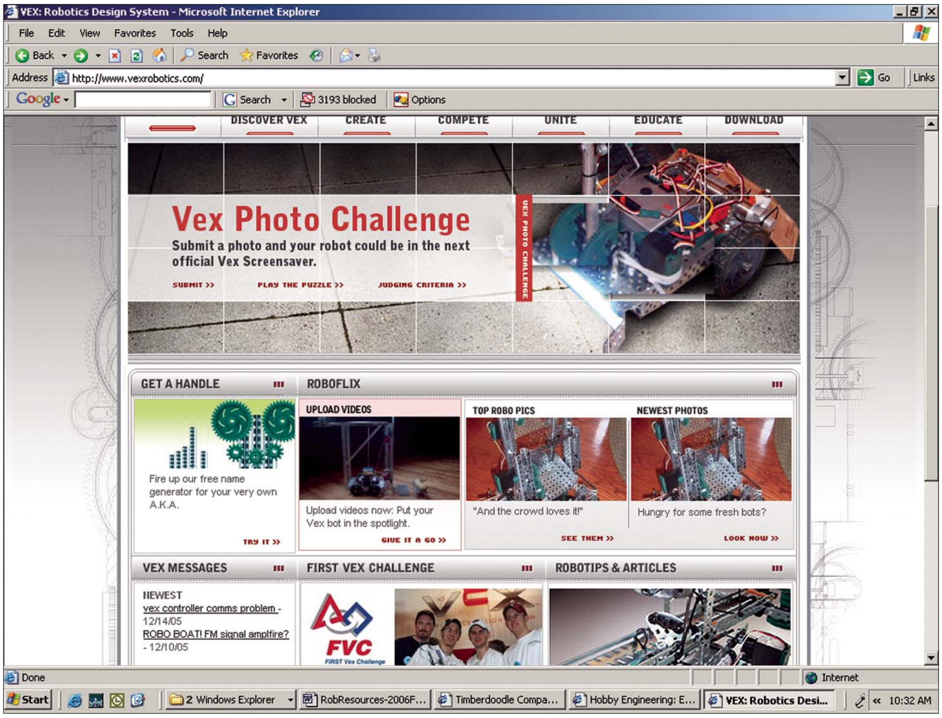


FIGURE 4. VEX Robotics uses Erector set concepts, updated for the express purpose of building a desktop robot.

Only Toys
www.onlytoys.com

Only Toys carries metal Erector sets; most are for building vehicles, and some (like the Steam Engine) are quite elaborate. The company also sells Rokenbok radio controlled toys.

Reid Tool & Supply
www.reidtool.com

Industrial supply source, including 80/20, Inc., aluminum extrusions.

Target
www.target.com

Retail stores and online site. Both offer great deals in clearance items. Make it a habit of regularly checking the website for clearance items.

Timberdoodle
www.timberdoodle.com

Timberdoodle specializes in home education products. They offer a good selection of Fischertechnik kits at good prices. Also sells Capsela, K'NEX, and electronics learning labs. Be sure to check their "swan gong" closeout deals.

VEX Robotics
www.vexrobotics.com

Makers of the VEX Robotics Design System. See also **RadioShack.com** for purchasing online. **SV**

ABOUT THE AUTHOR

Gordon McComb is the author of the best-selling Robot Builder's Bonanza, Robot Builder's Sourcebook, and Constructing Robot Bases — all from Tab/McGraw-Hill. In addition to writing books, he operates a small manufacturing company dedicated to low-cost amateur robotics, ***www.budgetrobotics.com*** *He can be reached at robots@robotoid.com*

GEERHEAD

by David Geer

Contact the author at geercom@alltel.net

A Fearsome Foursome of Recon Flyers

Paul Y. Oh — professor at Drexel University, Philadelphia and director of the Drexel University Autonomous Systems Lab (DASL) — builds flying robots together with his team of colleagues. Oh's field of robotics is referred to as indoor aerial robotics.

These robotic crafts can fly low to the ground in or around debris- or obstacle-laden territory such as buildings, caves, forests, or tunnels. Obvious potential applications for the military, Homeland Security, and public service needs include search and rescue operations, intelligence gathering, and mapping out unexplored environments for safety concerns before people are deployed.

Specific applications of the robots include security monitoring for venues, public places, and properties such as warehouses, stadiums, subway tunnels, and train stations, as well as any others you can imagine.

Background

You may recall that in the Arctic Tortoise GeerHead column a year ago, I noted that:

Prior to 2001, the Senate issued the Defense Authorization Bill for Unmanned Vehicles (UMVs); the Bill included the following statement. "It shall be the goal of the Armed Forces to achieve the fielding of unmanned, remotely controlled technology such that ... by 2015, one third of the operational ground combat vehicles of the armed forces are unmanned."

Congress is requiring a much faster pace of adoption for unmanned aircraft:

"Congress has mandated that by the year 2010, one-third of all deep strike force aircraft are to be unmanned (2001 Defense Authorization Conference Bill — H.R. 4205)," says Oh.

However we may feel about the war we are in today, I think most of us can agree that if we end up in another one down the road or are still in this one by 2010, so many unmanned vehicles amounts to the safety of troops who would otherwise have to be in those vehicles.

How Do We Get There?

Today's military class flying robots can hardly make muster if their exploits remind us of that ancient footage of the first flight attempts from around the time of the Wright brothers and before.

The primary obstacle to production ready robots is improving their reliability and autonomy. Not that it can't or won't be done. At the DASL Lab, Oh and his colleagues are addressing gaps between the current state-of-the-art in reliability and autonomy for these bots and the standard that they must meet in just five years.

"Recent events like 9/11, Operation Enduring Freedom, and Katrina, have made such focus even more urgent; unmanned aerial vehicles (UAVs) have proven to be force multipliers that provide first responders with real-time data for tasks like search and rescue, assessing structural integrity of buildings, and mission support," says Oh.

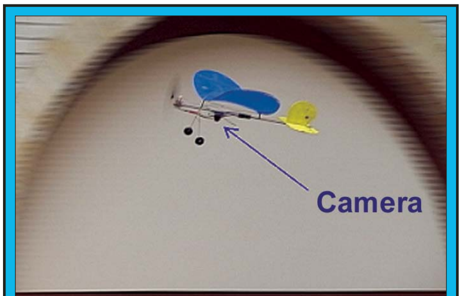
The DASL lab would like to produce aerial robots that soldiers can carry in their backpacks and deploy as needed.

"Several groups have developed bird-sized aircraft called micro air vehicles (MAVs)," says Oh. Some of these are fixed wing craft (airplanes), some are flapping wing craft (ornithopters — yes, that's a word), and some are rotary wing craft (helicopters). You can begin to see the breadth of the research underway to get these bots up and flying by 2010.

But, most of these UAVs aren't suited to near-Earth, indoor flight where technologies you would like to use such as wireless communications and GPS tracking are impaired.

Effective indoor flyers

Image of a CQAR unmanned vehicle with attached camera.



Shows size of li-poly battery powering the CQAR.

Shows size of toytronix motor used on CQAR.

have to be small enough to maneuver in tightly closed spaces. They have to fly slowly and safely to retrieve data and to avoid damaging things they might strike such as buildings. They have to use sensors to autonomously avoid such obstacles, as well. "Currently there is no vehicle that adequately addresses all four issues," says Oh.

DASL has been working on suites of sensors that meet the needs of near-Earth flyers since 2001. They have tested these sensors on a variety of flying vehicles. Types include the tethered, winged aerostats from the Low Elevation Aerial Platform (LEAP) project, the shrouded tandem rotorcraft from the Self Elevating Live Image Acquisition Platform (SELIA) project, the current Close Quarter Aerial Robot (CQAR) project flyers, and 3D acrobat fixed-wing (Blackhawk) aircraft.

LEAP — A Little Solution for Big Problems

The LEAPs are backpackable, quickly deployed, and very easy for soldiers to fly. They can climb 1,000 feet or about the height of a 70-story building in 10 minutes.

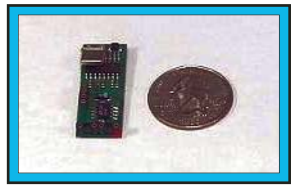
"It is essentially a hybrid weather balloon and kite equipped with a wireless camera to quickly provide aerial video," says Oh.

The tether lets you keep the LEAP in one spot for ongoing surveillance and monitoring. It can fly in one spot for days and deliver video constantly when mounted with an infrared camera. The tethered platform lets you control it without line-of-sight between you and the bot.

The kite aspect counters gusts of wind to keep the vehicle in position while the balloon provides elevation. The LEAP may sound like a cheap toy, but it's that affordability and dispensability that makes it viable and expendable to first responders who need an "eye-in-the-sky" in scenarios where it is prohibitive to send up more conventional craft.

"For example," says Oh, "natural disasters often result in crippled runways that prevent airplanes from getting to the site. In Katrina, helicopters were also a problem; when flying close to water, the resulting rotor wash can forcibly submerge and drown victims. LEAP is a low-cost and safe platform to address many of the issues that current aircraft have not adequately addressed."

Shows size of CQAR propeller.



Shows size of radio receiver used on CQAR.

CQAR Anyone?

CQAR is a fixed-wing platform that weighs 30 grams (about .066 lbs.). The CQAR was created to navigate in and around buildings. It can be used for recon or to gauge the structural integrity of a building before soldiers or first responders enter.

The CQAR flies at only 4 to 5 mph so that it has enough time not only to sense obstacles, but also to navigate around them before colliding.

Carbon fiber tubing and balsawood were used to build the vehicles frame. "The resulting prototype, with electronics, weighed 30 grams (about the weight of three quarter coins) and could fly at speeds of 5 mph while carrying a 10 gram wireless camera," says Oh.

Blackhawk Up

The famously named Blackhawk is a DASL lab aircraft intended to surveil and recon tunnels and caves. Unique among its properties is the ability to hover despite being a fixed wing vehicle.

"This is achieved through a high thrust-to-weight ratio that enables the platform to transition from cruise mode (i.e., wings parallel to the ground), through the stall regime of conventional fixed-wing aircraft and into hovering mode (i.e. longitudinal axis of fuselage is vertical)," says Oh. Unique to the crafts construction is a Depron foam core laminated with 2 oz per inch of carbon fiber.

The Blackhawk can soar steadily at speeds up to 40 mph. It is capable of long flight times thanks to its fixed-wing configuration. The Blackhawk weighs about a pound and can additionally carry a 100-gram payload such as a camera while hovering.

SELIA We Love You

The last of the four, SELIA is a shrouded tandem rotor vehicle with an oval frame that can fit in a soldier's back-

MORE SPEC-CIFICALLY

While LEAP is primarily enabled by a transmitter, the CQAR uses its onboard control system to monitor sensor output. When moving in on an object to its left, the control system sends a signal to the rudder to make a right turn, for example.

"LEAP uses a 72 MHz transmitter for the camera's pan and tilt servos and a 2.4 GHz wireless camera," says Oh.

In landing mode, the control system monitors sensing and sends signals to the elevator to control the CQAR's rate of descent. CQAR uses a PIC16F84 microcontroller for its onboard control system. "The software embedded on the controller was written in C. Under autonomous mode, all communications are done onboard the aircraft. A 900 MHz wireless camera is used to acquire surveillance video," says Oh.

As for the Blackhawk, a human operator can switch from manual to autonomous control and back again as needed. In its autonomous mode, this flyer also has an onboard control system — to control the rudder and elevator for hovering attitude.

The Blackhawk control system consists of a PIC16F87 microcontroller and a MAX232 chip to communicate via RS232 with the attitude sensor, according to Oh. This flyer's software, used to read data from the sensor and to control the servos accordingly was also written in C. A 2.4 GHz wireless camera is used to acquire surveillance video.

All of these robots are reprogrammed by first modifying the C code and then compiling the software using PIC C. "This generates the hex file which is burned onto the micro using a PIC programmer," says Oh.

Year 1 of the competition focused on sensing (computer vision) and human-robot interaction (teleoperation) and hence a simple and safe-to-fly platform (a blimp) was used. Drexel team member Jason Collin inspects the robotic blimp.

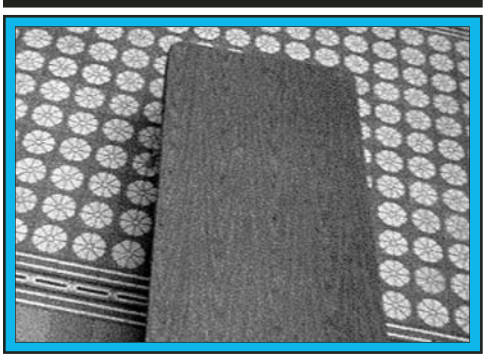
pack. It can hover and stare at whatever it is surveilling. "SELIA can ascend a building face and peer through windows or openings for search and rescue, reconnaissance, or structural integrity assessment tasks," says Oh.

The vehicle's carbon-fiber shroud protects its two rotors so that the vehicle can survive minor collisions. "Its tandem rotor configuration is equivalent to the tail rotor of a helicopter and acts to counter the motor torque," says Oh. The rotors provide the lift necessary to carry a 15-gram wireless camera in addition to its own weight.

No One Can Compare

So, how do these cheap, slight, and dispensable flying robots stack up against the big boys? While helicopters can retrieve similar intelligence data as the LEAP robot, the natural disasters and terrorist attacks that cripple roads, bridges, and runways denying access

Recreation of image taken by camera on CQAR.



Drexel team members William Morgan (front) and Jason Collins (back) tuning the computer vision algorithms to autonomously navigate the aerial robot.

for such vehicles don't stop the LEAP from doing its job.

Satellites can also acquire this intelligence but it takes longer to reprogram them to do a fly-by than it takes to deploy a LEAP to get the same information.

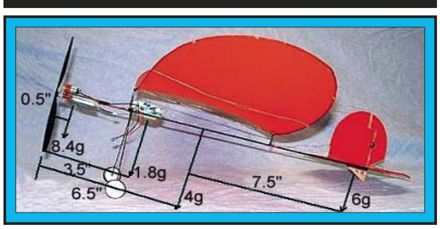
Other "bird-sized" flyers like the CQAR are made entirely of fiberglass or carbon fiber, adding unnecessary weight. As such, they must fly above 15 mph to stay aloft. "Utilizing lightweight materials such as Mylar, balsawood, and carbon fiber tubes, the CQAR prototype is capable of speeds well below this," says Oh.

The Robotic Foursome

The LEAP robot is teleoperated. It can do its job from a distance. This little spy can provide its human operator with live video streams. An operator can work with the robot's camera to capture a specific image from among the camera's field-of-view.

CQAR uses sensing that mirrors that of honeybees. Honeybees use

This is a 27-gram, 19-inch wingspan aircraft that can fly as slowly as 4 mph and carry a 15-gram wireless camera. It demonstrates that aircraft can be designed to fly in closed quarters.



RESOURCES

LEAP video
http://prism.mem.drexel.edu/projects/kite/index.html
(some software plug-ins may be required for viewing)

CQAR video
www.pages.drexel.edu/~weg22/CQAR.html

How to build your own CQAR
www.pages.drexel.edu/~weg22/CQAR.html

Blackhawk video
www.pages.drexel.edu/~weg22/fwHovering/fixedWingHovering.html

SELIA video
www.pages.drexel.edu/~jg39/Pages/Senior%20Design/senior_design.htm

The Drexel Autonomous Systems Lab homepage
http://dasl.mem.drexel.edu
You will find information about the DASL flying robots and upcoming demonstrations.

Intro to Professor Paul Y. Oh
www.mem.drexel.edu/pauloh.html

something called optic flow to determine object avoidance and landing. Optic flow can be defined as the motion of texture in the field-of-vision relative to the insect's flying velocity. Closer objects have higher optic flow. So, the honeybee avoids areas of high optic flow. The honeybee also lands using optic flow because the optic flow of the ground stays constant.

"With an optic flow sensor and these flight stratagems embedded onto the onboard control system, the CQAR robot was able to demonstrate autonomous collision avoidance and landing. To the best of our knowledge, these were the first optic flow based autonomous maneuvers performed indoors," says Oh.

The Blackhawk can hover with the help of an operator who must constantly work the four channels of the radio transmitter. "However, retrofitting Blackhawk with a sensor that detects the aircraft's attitude, we were able to develop an onboard controller to automate the hovering flight mode. To the best of our knowledge, this is the first autonomous hovering of a fixed-wing unmanned aerial vehicle ever performed," says Oh. SV

Rubberbands and BAILING WIRE

by Jack Buffington

Inverse Kinematics

How to Put Your Robot's Foot in Its Mouth (With Precision)

If you are interested in building robots that are a bit more anthromorphic than average, inverse kinematics is a concept that you will eventually have to bend your mind around. There are two methods of getting a robot with appendages to move. These methods are called forward kinematics and inverse kinematics. If you are familiar with Robo-One robots then you will have seen forward kinematics in action in most of them. Forward kinematics is the sort of control that you do when you specify joint angles ahead of time for your robot's legs or arms to get its foot or hand to the proper location.

Many Robo-One robots are controlled by storing the positions of all of the servos for a given pose. A computer moves the robot by interpolating between the various saved poses. Inverse kinematics lets you specify the position of the foot or hand and your software takes care of figuring out all of the joint angles. Using inverse kinematics allows you to move your robot in any manner that you would like at run time without having to plan your poses or having to store information about servo positions.

How is this more useful than forward kinematics? It allows you to deal with unanticipated situations more easily. Let's say that you wanted to have your robot walk along rough terrain and you would prefer to not stress its motors if the foot steps on a rock that is higher than the ground; with inverse kinematics, you could recalculate the path that each foot needed to take so that you would end up loading each leg with an even amount of the robot's weight.

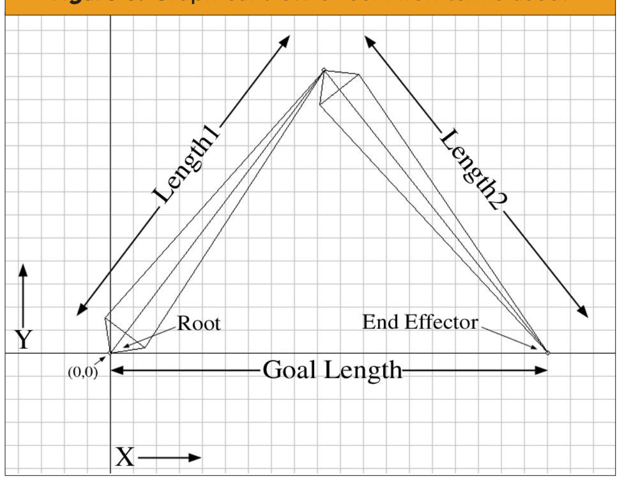
In this column, we'll look at two different ways to compute inverse kinematics. The first case is a simplified case where there are only two joints that move. The second case allows for any number of joint angles to be computed. Let's go over a few terms that will be used in this column. If you are familiar with computer animation, you will already know these terms. In animation, you deal with concepts called bones. Like real bones, these are rigid structures that are connected with joints. A series of connected bones are called a chain. The root of a chain is the joint that may rotate but always stays in the same physical location. If you were describing a leg, the root would be the hip. The other end of the chain is called the end effector. Inverse kinematics (IK) uses a goal point in its calculations. The goal is where the IK calculation tries to position the end effector. Each joint will have a preferred angle so that it doesn't try to bend backwards.

Let's say that you have a robot that has a leg with two bones in it that are both 10 cm long. In this case, you will be able to achieve any position that is zero to 20 cm away from the hip if your robot could fold its leg up completely and also make it be perfectly straight. When calculating your IK solution, you will want to verify that the goal you are trying to achieve is actually reachable. To do this, use the Pythagorean theorem to find the length from the root to the goal. We'll call this the goal length. If the goal is not reachable, you will have to figure out a way to deal with that situation.

Usually, you will be able to reach most of the points near the root and everything up until the combined length of the bones. If you are trying to achieve a goal point that is farther away than the combined length of the bones, then you might want to just make the chain of joints line up in a line and cause the end effector to point in the direction of the goal. That way, when the goal moves back into a place where the end effector can reach it, the joints won't have to rotate violently to get there.

Once you have determined that you can actually reach the goal, you are ready to figure out your joint angles. With two bones, this process is fairly simple. You will just need to figure out the correct knee or elbow angle that will cause the distance from

Figure 1. Graphical view of common terms used.



```
int length1Squared = length1 * length1;
int length2Squared = length2 * length2;
int goalLengthSquared = goalLength * goalLength;

int X = length1Squared - length2Squared + goalLengthSquared;
X  /= (2 * goalLength);

int Y = -(sqrt(length1Squared - (tempX * tempX)));
```

Figure 2. Code to find the intersection of two circles.

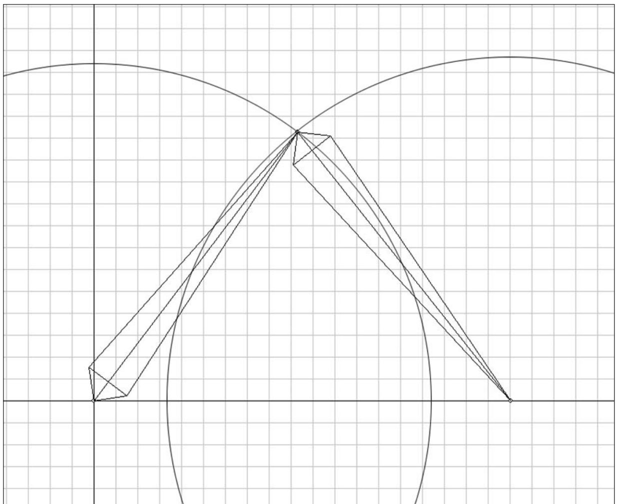


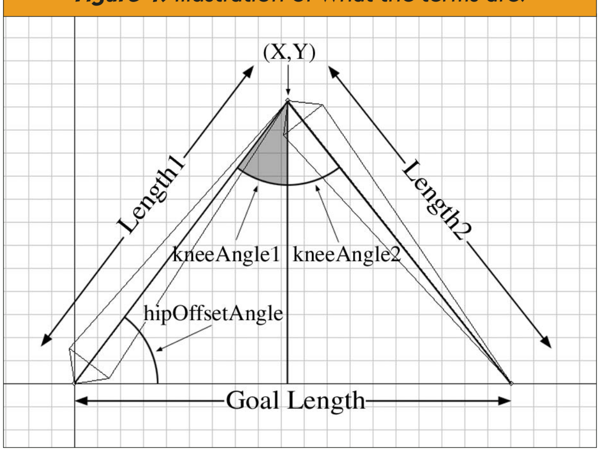
Figure 3. The two circles that intersect.

the root end effector to match the goal length. You can calculate this by doing either of two things.

The first way is to simply make a lookup table that says what the angle should be for every goal length that the chain can achieve. This is by far the fastest to compute at run time but also requires a lot of program space. The other way is to assume that the root and the goal lie along the X-axis of an imaginary graph. You will need to provide your program with information about the lengths of the "bones." The program will use this information to find the intersection of two circles that radiate out from the root and the goal with radiuses the lengths of bones. Take a look at Figures 2 and 3 for a code example and a diagram of what this code represents.

As you can see from the code and diagram, we are putting the centers of the two circles on the line where Y equals zero. The root has an X coordinate of zero and the goal has an X coordinate that equals the goal length. This simplifies our calculations. It is often the case that two circles intersect in two points. If for some reason you wanted to know the other point, just simply take the negative of the Y coordinate. That point is not needed for this application though, so we can just ignore its existence. Now we have the coordinates of the knee but still need to figure out a couple angles. We'll call these angles kneeAngle1, kneeAngle2, hipOffsetAngle, and hipAngle. Let's figure out kneeAngle1 first.

Figure 4. Illustration of what the terms are.



For that angle, we take the arcSin of X/length1. Just as a quick review, taking the sine of an angle returns the ratio of the lengths of the opposite side of a triangle and its hypotenuse. Arcsine is the opposite. If you give it the ratio of the lengths of the opposite side and the hypotenuse, it returns an angle. At this point, we have a partial answer for the knee angle.

The next step is to find the other part of the knee angle by taking the arcsine of the goalLength minus the knee's X coordinate divided by length 2. Add kneeAngle1 to kneeAngle2 to get the final result. Take a look at the code listing in Figure 5 to get a better idea of how this works. We now have the correct angle for the knee.

Let's look at what the hip should be doing. This one is easier to calculate. If we are using 360 degrees to represent all of the angles in a circle then we can take 180 minus the first angle that we found for the knee to get the hip offset angle. This works because all of the angles in a triangle add up to 180 degrees.

You now have enough calculated to move the foot in a straight line up and down. This isn't too useful though, since your robot would only be able to march in place. Let's calculate one more angle that will allow the foot to achieve the proper location. We'll add this angle to the hip offset angle. Figure 6 shows what this angle is. To find this angle, just take the arcsine of the X axis and the goal length to get the angle that you add to the hip offset angle. You can now successfully place the end effector exactly where you need it to be.

We just discussed a common situation where there were only two joints that needed to be manipulated. What if you need to use more than two joints but would still like your end effector to end up in the right place? It becomes a lot more difficult to calculate the correct position in one pass as we did with two joints, so this method of calculation uses successive approximation to arrive at an answer. This method will work for most leg or tentacle situations.

The way that this strategy of calcu-

Figure 5. Code to figure out how to position the knee and hip for a leg.

```
kneeAngle1 = asin(kneeX/length1);
kneeAngle2 = asin((goalLength - kneeX)/length2);
kneeAngle = kneeAngle1 + kneeAngle2;
hipOffsetAngle = 180 - kneeAngle;
```

lating inverse kinematics works is that once again you will figure out the goal length for the chain. This method takes into account a default angle for each of the joints. Figure 7 shows how a chain will stretch using this method. If the goal length is greater than the default length from the root to the end effector, then the angles of the chain will become more obtuse. If the goal length is less than the default length from the root to the end effector, then the joint angles will become more acute.

Let's look at the situation where the angles need to become more obtuse. Each joint will move proportionally from its default angle to being perfectly straight. If a joint is far from straight in its default position, then it will move a lot as the chain is stretched. If the angle is acute, it will not move very much when the chain is stretched. Take a look at Figure 8 to see how this works.

You will use a binary search technique to adjust the joint angles until the chain length is within a certain tolerance or a certain number of iterations have occurred. After each time the joints are adjusted, your program will figure out what the length from the root to the end effector is at that point and compare it to the goal length. Once a satisfactory arrangement of the joints is found, then the root joint will receive an offset to move the end effector into the desired location just like we did with the IK code for two joints.

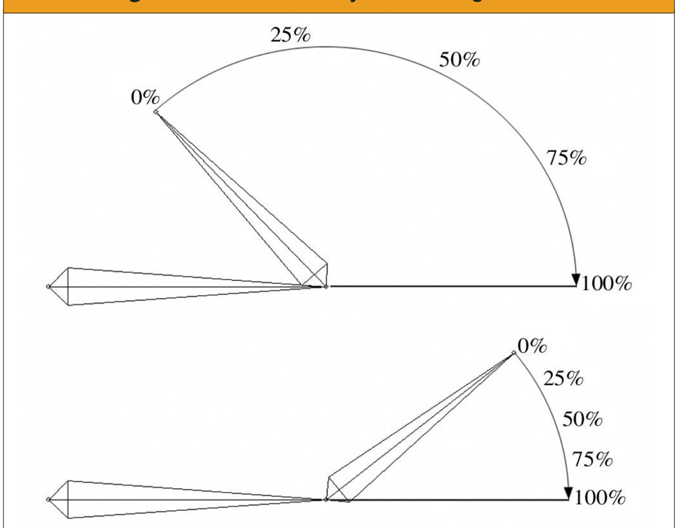
Wrap Up

Two different manners of figuring out inverse kinematics were shown in this column. The first one is fairly fast to calculate and works nicely for most situations. For times when you need additional joints, the second approach works well. It isn't nearly as fast of a strategy so it should be avoided if possible. It is also worth noting that any solution that the second approach could achieve could also be achieved by using one active joint at the root and one active joint in the chain. The rest of the joints could be mechanically linked to achieve the same result.

That approach requires 0% of your processing power for all of the passive joints and you can simply make a precalculated lookup table that takes the active joint's angle and returns length of the chain to help you figure out the proper angle to drive the active joint to.

Legs and arms are something of a holy grail for robotics. Most hobby robots don't stray too far from being two-wheeled robots that do little more than drive around and avoid things. If you are looking to take your next project to a whole new level, consider putting some legs on it. Hopefully, this month's column has given you a new understanding of how you can get started. **SV**

Figure 8. How much a joint is straightened.



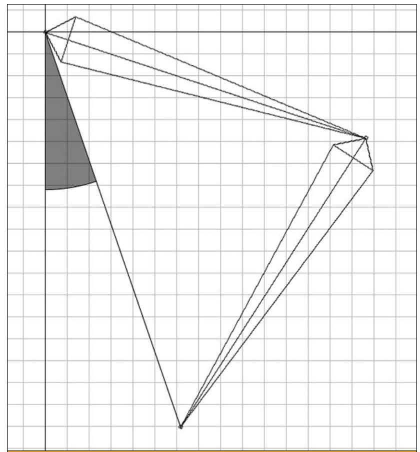


Figure 6. The angle to add to the hip offset angle to place the foot in the proper location.

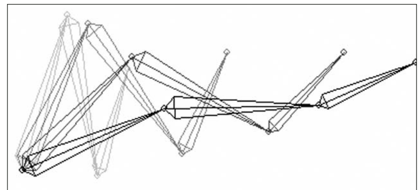


Figure 7. A chain of three joints being stretched out.

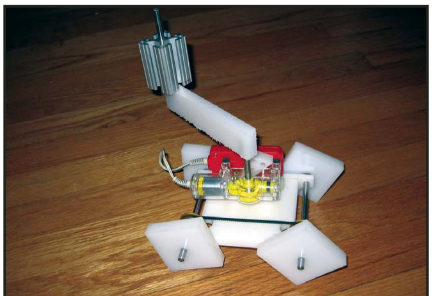
Robytes

by Jeff Eckert

Are you an avid Internet surfer who came across something cool that we all need to see? Are you on an interesting R&D group and want to share what you're developing? Then send me an email! To submit related press releases and news items, please visit www.jkeckert.com

— Jeff Eckert

Locomotion on Square Wheels



This vehicle travels on square wheels, driven by weight shifts. Photo courtesy of Global Composites.

You probably never expected to see a practical vehicle that employs square wheels, but a patent-pending prototype seems to have potential for robotics, micro machines, toys, and so forth. The brainchild of Jason Winckler, of Global Composites, Inc. (**www.globalcomposites.net**), it basically uses gravity for propulsion. The wheels are mechanically connected, with their rotational orientation offset from each other by 22.5° (one fourth of the 90° of movement from one flat side to the next). A motor rotates a weight above the vehicle and the weight shift sequentially drives each wheel so that the device moves ahead. Reversing the rotational direction also reverses the vehicle's direction of travel.

According to a company representative, "For use in micro machines or MEMS applications, one of the key benefits is that the motor and gearing moving the shifting weight is all in a plane parallel to the motion surface. No right-angle gearboxes are required. The connection between the two axles can be accomplished by simple linkages."

Although the weight/gravity configuration provides drive power in this version, the company already foresees versions that employ aerodynamic, hydrodynamic, magnetic, electromagnetic, and electrostatic forces. Being independent of the car's mass, these approaches could provide faster and more powerful devices, and the motor could be eliminated in some cases. Reportedly, Global Composites has already had discussions with several companies who are interested in licensing the concept.

ASIMO Now a Bartender

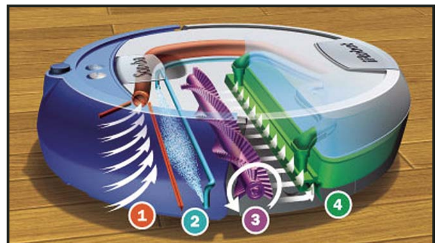
Honda's upgraded ASIMO features a range of new functions. Photo courtesy of Honda Motor Co.

In December, Honda unveiled a new version of its ASIMO humanoid robot with enhancements that allow it to better interact with people, allowing it to act as a receptionist, guide, or delivery boy. This has been achieved by providing him with the ability to recognize the surrounding environment through visual sensors, a floor surface sensor, an ultrasonic sensor, and the "IC Tele-interaction Communication Card," which is held by the person with whom ASIMO will interact.

With this card, ASIMO can recognize the location and identity of any person in a 360-degree range. Through the coordinated use of its eye camera in the head and the force (kinesthetic) sensor on its wrists, ASIMO can give and receive objects such as trays and other items. Furthermore, by using the force (kinesthetic) sensor, ASIMO can hold your hand and walk in sync with you.

The previous version could move at only 3 km/hr, but this one can double that. It can also run in a circular pattern without tipping over, as the bot's center of gravity can be tilted toward the center of the circle. The new ASIMO will begin operating in Honda's Wako office building this spring and eventually will be available for lease.

Robotic Mop



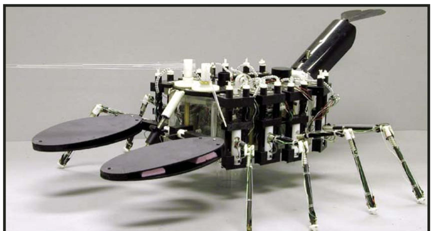
iRobot's newest consumer product preps, washes, scrubs, and dries floors simultaneously.

Just when you got used to the Roomba robotic vacuum cleaner concept, iRobot Corp. (**www.irobot.com**) has followed it up with the Scooba floor washing robot. It simultaneously preps, washes, scrubs, and dries hard floors automatically. Unlike a conventional mop that ends up using dirty water from a bucket, Scooba uses only fresh water and cleaning solution, sucking up the dirty water as it goes, and one tank will clean about 200 sq ft in four passes. The thing can also suck up wet spills in addition to removing normal floor dirt, and it is said to be safe for all sealed, hard surfaces, including wood and tile.

Scooba will be available at retail outlets soon, perhaps by the time you read this. It's priced at $399.99, and a five-pack of cleaning fluid (specially formulated by Clorox) will set you back $25. A range of accessories are available, including an infrared virtual

wall that allows you to control what areas will be mopped.

Not Available at Red Lobster



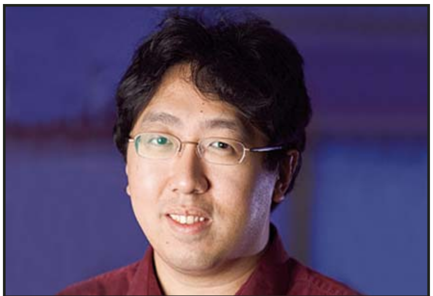
This eight-legged autonomous lobster is under development at Northeastern University. Photo by Jan Witting, courtesy of Northeastern.

One of the strangest looking bots I've seen lately is an eight-legged ambulatory vehicle based on your average Maine lobster. Developed at Northeastern University's Marine Science Center as part of the Biomimetic Underwater Robot Program, it is intended for autonomous remote-sensing operations in rivers or the near-shore zone ocean bottom with the ability to adapt to irregular bottom contours, current, and surges.

The Center has also developed a legless, undulatory critter that is based on the lamprey and intended for similar purposes but in water columns of great depth. They employ a common biomimetic control, actuator, and sensor architecture and are based on modularized components to minimize cost. For detailed information, including animations, visit **www.neurotechnology.neu.edu/** Bring your own drawn butter.

Algorithm Improves Monocular Vision

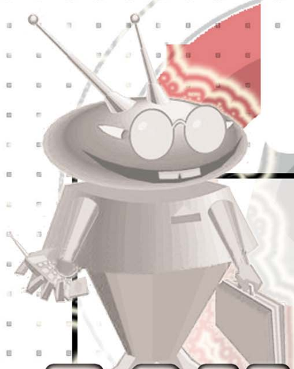
It has been pretty much assumed that accurate depth perception requires two eyes, be they human or otherwise. However, some devices are too small or under extreme cost constraints that make stereo vision impractical. It appears that Prof. Andrew Ng, with the assistance of some graduate students at Stanford University (**www.stanford.edu**), has come up with a package of computer algorithms that allow robots to fairly accurately guess distances using single still images.



Prof. Andrew Ng has developed improved depth-estimation algorithms. Photo courtesy of Stanford University.

The software employs "cues" in the images, including texture variations, edges, and the amount of haze to generate the estimates. It breaks the images into sections and analyzes them both individually and in terms of how they relate to adjoining sections, and it also varies the magnification to make sure it catches all the image details. Reportedly, the result is that robots using the software — in both indoor and outdoor locations — have been able to judge distance with an average error of about 35 percent. That may sound like a substantial error, but Ng points out that the robot would perceive a tree that is 30 ft away as being anywhere between 20 and 40 ft away.

If the bot were traveling at 20 mph and processing images 10 times per second, that would provide it with enough time to adjust its path and avoid the tree. And the software can operate at distances up to 10 times as great as can be handled by stereo vision algorithms. Obviously looking toward further refinements, Ng noted, "I'd like to build an aircraft that can fly through a forest, flying under the tree canopy, and dodging around trees." I think we'd all like to see that. **SV**



Our resident expert on all things robotic is merely an Email away.

roboto@servomagazine.com

Tap into the sum of *all human knowledge* and get your questions answered here! From software algorithms to material selection, Mr. Roboto strives to meet you where you are — and what more would you expect from a complex service droid?

ASK MR. ROBOTO

by **Pete Miles**

Q. Is there any easy way to accurately transfer a drawing to a piece of plastic? I have a bunch of parts I made with Autocad that I want to cut out of plastic. Simple parts are easy to redraw on the plastic sheets, but I have a bunch of parts with lots of curves and holes that I need to align, and it takes too much time to copy them accurately enough to make them right. Do you know of an easier way to do this?

— Samuel Dodge

A. The easiest way to do this is to use good, old-fashioned rubber cement. What I do is make a full size drawing of the part on a plain piece of paper. This can either be done by hand or by printing the CAD file with a 1:1 scale. If you made your drawing by hand and need multiple copies of the same part, then photocopying the part drawing will save you a lot of time having to redraw the same thing over and over again.

With some scissors, trim the drawing so that there is about an inch of extra paper all the way around the part. Then place the part drawing on the plastic that is going to be cut. I use a pencil to make a few marks on the plastic so that I know where the paper belongs. I then remove the paper and brush a thin coat of rubber cement on the back side of the paper, and a thin coat on the plastic sheet where I marked the paper location. Then starting with one side of the paper, I slowly rub the two sticky sides together. Make sure that you work from one side to another or you will get some air bubbles or wrinkles. Bubbles or wrinkles will distort your part.

After about 10 minutes, you are ready to start cutting out your part. You can use either (or a combination) a coping saw, scroll saw, drill, and disk sander to shape the part. All you have to do is cut along the lines on the paper. When it comes to drilling holes, make sure that you add a set of cross hairs to the center of the hole. It is a lot easier to align a drill bit to a cross hair than trying to center the drill bit in a hole by eye.

After the part is finished, just peel the paper off the part. Most of the time, the rubber cement will stay on the paper, thus making it really easy to use. Sometimes there is a little bit of rubber cement still on the part, but this can be easily rubbed off. This method works great on all smooth plastics and metal surfaces. This process also works with wood parts, but only put the rubber cement on the paper side. Don't put the rubber cement directly on the wood. By doing it this way, you minimize the chance of getting any rubber in the pores/grain in the wood which is difficult to remove without sanding the surface.

Figure 1 shows a drawing of a robot body part to be made from 1/8 inch thick Sintra (Expanded PVC). Figure 2 shows the trimmed printout glued to a scrap piece of plastic. Figure 3 shows the part after all the holes were drilled and the perimeter cut to shape. Figure 4 shows the final part with the paper template removed. Using this approach makes reproducing parts quick and easy. It

Figure 1. Drawing of a robot body part to be cut out.

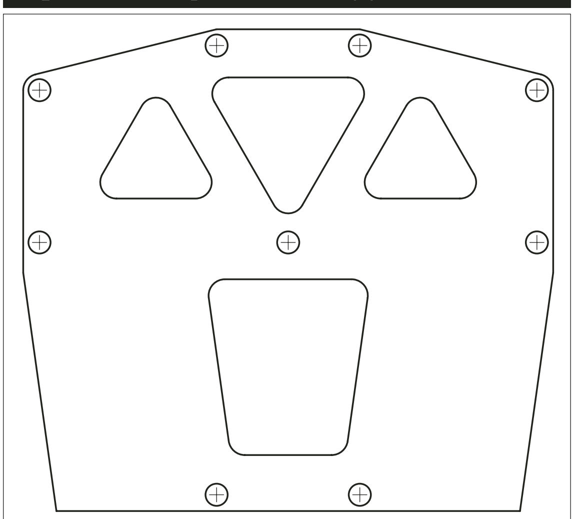


Figure 2. Trimmed printout glued to a workpiece.

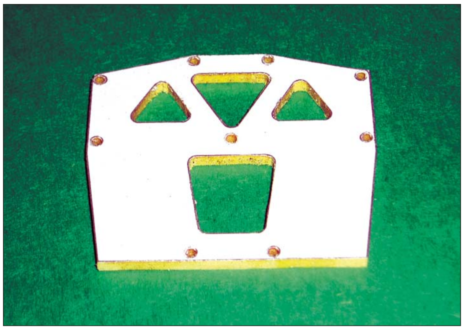


Figure 3. Holes drilled and perimeter hand cut to the lines on the glued template.

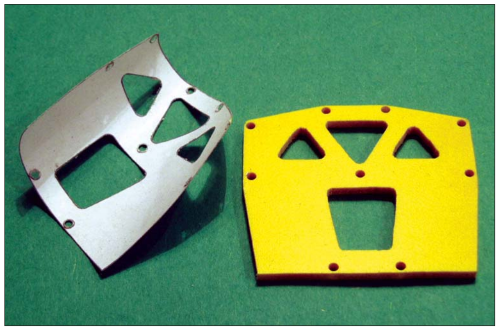


Figure 4. Finished part with the template removed.

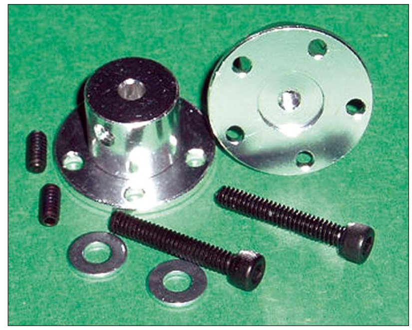


Figure 5. Lynxmotion mounting hub, HUB-7, with mounting hardware.

took less than 20 minutes to print the full scale drawing and cut out the part.

Q. I have some foam model airplane wheels that I got from a model store that I want to use on my robot, but they don't have any good way to attach to a motor shaft. I have tried hot glue, but it keeps coming loose. Do you have any suggestions how I could use these wheels?

— Ted Hwong

A. Attaching off-the-shelf wheels from the R/C plane and car community has posed many challenges to the robot building community. This is mainly because they assume that the wheels are going to be either free spinning or attached to their specific drive hubs. If you have a lathe at home, you can easily make your own drive hubs that will fit perfectly between the motor and wheel. Unfortunately, most of us don't have a lathe and would prefer an off-the-shelf approach to doing this.

One company — Lynxmotion (**www.lynxmotion.com**) — has made several different types of mounting hubs that will work for many different types of wheels. The Robot Store (**www.robotstore.com**) and Robot Market (**www.robotmarketplace.com**) also offer several different types of hubs that can be used with these foam wheels.

The 3 and 4 mm mounting hubs from Lynxmotion (part numbers HUB-7 and HUB-6, repsectively) are ideal for mounting foam wheels to motors. The mounting hub has a 3 or 4 mm diameter bore for directly mounting onto the drive/motor shaft, and has a #6-32 threaded hole for using the supplied set screw to lock the hub onto the drive shaft. Figure 5 shows this mounting hub.

This hub has two different methods for attaching to a wheel. The hub's flange has a set of five 0.09 inch diameter thru holes. A set of #4 (or smaller) screws can be used to screw the hub directly to the side of the wheel (see Figure 6). If you use a #4 screw, you may need to open up the diameter of the holes so the screws will slip easily through. An 1/8 inch diameter drill will make the hole about 0.010 inches larger in diameter for a #4 screw.

The second method uses a #5-40 screw as a secondary axle for mounting the wheel to the hub. A #5-40 screw is passed through the center of the wheel and is screwed into the tapped hole in the center of the hub. A small washer should be placed between the screw head and the side of the wheel, and the screw is tightened down. Figure 7 shows an exploded view in how to assemble the hub to the wheel and motor (the foam tire is not shown here). Figure 8 shows all the parts assembled together. Figure 9 shows the mounted hub with the foam tire.

There are a couple things to keep in mind when using the center screw approach for mounting any wheel to any hub. Reversing the motor has the tendency of loosening the mounting screw, and too much torque on the wheel could cause the wheel to slip/spin on the screw. If your wheel starts to slip relative to the hub, you can either tighten the screw, add a lock washer between the wheel and the screw head, or place a screw through one of the holes on the flange into the side of the

Figure 6. Bolting the Lynxmotion mounting hub to the side of the wheel.

Figure 7. Exploded view showing how to assemble the Lynxmotion mounting hub to a foam wheel mount and motor.

Figure 8. Completed assembly process (less foam tire).

Battery Type	Rechargeable	Energy Density Whr/kg	Cell Voltage
Lead Acid	Yes	40	2
NiCd	Yes	60	1.25
Rechargeable Alkaline	Yes	80	1.5
NiMH	Yes	90	1.25
Lithium-Polymer	Yes	120	3.6
Alkaline	No	130	1.5
Lithium-Ion	Yes	140	3.6
Lithium	No	300	3

Table 1. Common Battery Technology Energy Densities.

Figure 9. Hub mounted to a foam wheel.

wheel. The screw will prevent the wheel from spinning relative to the hub.

Q. What is the best energy to weight ratio for batteries?

— Stanley Gracey

A. The energy to weight ratio for batteries is defined as Watt-Hours of stored energy per kilogram (Whr/kg) of weight. The energy density of batteries are different for different types of chemistries. Also, for a specific chemistry, the energy density can have a fairly large range of variations depending on the physical size of the battery housing, and manufacturing methods. Table 1 shows a list of some of the more common types of batteries that we are exposed to on a day-to-day basis. The values presented in this table are typical values that can be expected.

Across the board, the lithium-based battery chemistry has the highest energy densities. Higher energy density doesn't necessarily mean they will be the best battery for your needs. Generally, lithium-based batteries are smaller in size, and have a lower overall energy capacity than the larger lead acid, NiCd, and NiMH batteries. Lithium batteries are the most expensive batteries on a pound per pound basis. If small, lightweight batteries are not a requirement, NiMH batteries usually have the best price for performance rating, and can tolerate a lot of charging and discharging abuse. **SV**

NEW PRODUCTS

ACCESSORIES

Smart/Fast Programmable NiCd/NiMH Charger

Cell-Con Incorporated, Exton, PA has announced the availability of a new 65 watt line of cost-effective, factory programmable NiCd/NiMH battery chargers for packs consisting of 3-20 cells (capacities up to 12 Ah).

Chargers can be programmed to terminate charging via –dV, dT/dt, and 0dV detection methods. Each phase of charging (initialization, fast, and top-off) has a programmable, safety back up timer.

Input is 90-264 VAC/47-63 Hz with charge rates between 1.8A and 4.5A, depending upon exact model. Safety approvals include UL (pending) and CE 60601-1. The unit contains an integral LED status indicator.

Available in two-wire and three-wire configurations (three-wire for thermistor interaction), the dimensions are 5.30" x 3.15" x 1.75" and it weighs .75 pounds.

The charger is available for immediate delivery. Single piece pricing starts at $125 with multiple, standard price breaks.

For further information, please contact:

Cell-Con Incorporated
Tel: **800 • 771 • 7139 ext. 210**
Website: **www.cell-con.com**

The ETH32

Winford Engineering has just released the ETH32, a general-purpose I/O device which communicates over Ethernet. This device is ideally suited for remote data acquisition or device control; all that is needed is a connection to an Ethernet network. The Ethernet connectivity and TCP/IP communication provide a great deal of flexibility, allowing the ETH32 to be located a long distance away from the PC, if so desired.

The ETH32 includes a variety of useful features for data acquisition, monitoring, and control purposes. It includes a total of 34 I/O lines, some of which can be used for special features. In addition to digital I/O, the ETH32 offers analog inputs, digital counters, and pulse width modulation (PWM) outputs. The ETH32 supports up to five simultaneous TCP/IP connections, allowing multiple computers to communicate with the ETH32 device at one time.

One of the powerful and useful features of the ETH32 is its event monitoring capabilities. In a nutshell, event monitoring allows the ETH32 to monitor different input signals and send a notification to your application when that signal has changed or met your criteria. Since the monitoring is constantly performed directly by the ETH32, it provides a much better alternative to polling over the Ethernet connection. It provides faster response, is very efficient with network bandwidth and CPU resources, and is typically much easier to implement in applications. Event monitoring capabilities are included for digital I/O ports, analog channels, and digital counters.

The ETH32 offers full-featured software support and comes with software libraries for both Windows and Linux, making it very simple to use the device. Support is included for Microsoft .NET languages, Visual Basic, C, and C++. In addition, the protocol used over the TCP/IP socket is documented for those who want to do the network programming themselves or are using an unsupported platform. Libraries, documentation, and sample programs are freely available for download on Winford's website. The ETH32 retails for $225 in single unit quantities.

For further information, please contact:

Winford Engineering
4169 Four Mile Rd.
Bay City, MI 48706
Tel: **877 • 634 • 2673**
Website: **www.winford.com**

Show Us What You've Got!

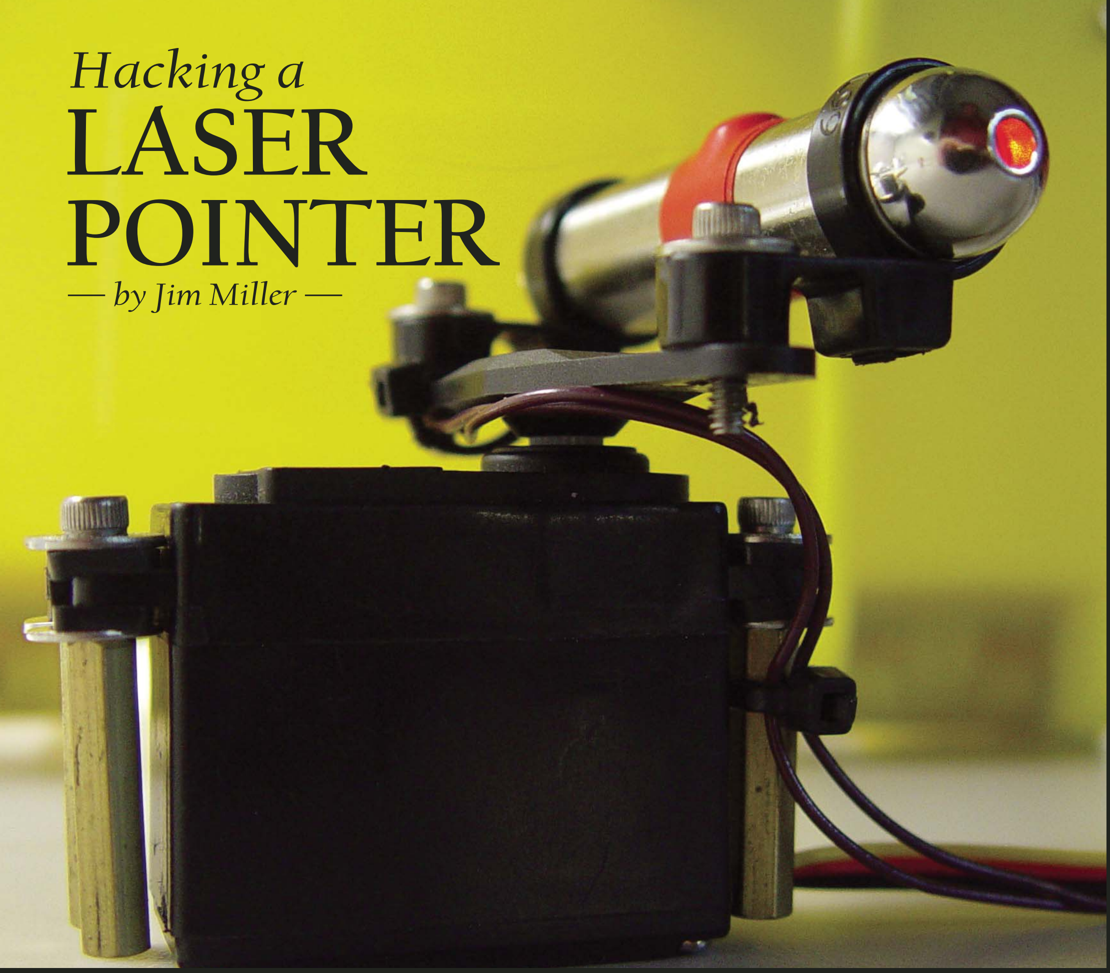
Is your product innovative, less expensive, more functional, or just plain cool? If you have a new product that you would like us to run in our *New Products* section, please email a short description (300-500 words) and a photo of your product to:

newproducts@servomagazine.com

Hacking a

LASER POINTER

— by Jim Miller —



One day I decided I wanted on and off control of a laser pointer. The pointers are very easy to mount on a servo, they're light, and great for sighting. I poked around for a cheap key chain laser for several weeks. I finally found one for about six bucks at 7/11. It would have to do and was about as cheap as I had seen anywhere else. Going to work, I immediately pulled the batteries out; 3 x 1.5V = 4.5VDC, so five should probably be okay, right?

I CUT THE BATTERY CAGE OFF THE POINTER AND WIRED IT UP TO A POWER SUPPLY. I TURNED IT ON AND IT SHONE BRIGHTLY FOR A SECOND THEN NEVER AGAIN DID IT GLOW. So, like any determined builder, I went to 7/11 and got another pointer. This time I'd just pull the diode out. Of course! I should have done that the first time. A laser pointer case is all epoxy and is designed to stay that way. About five pointers later, I had given up. There are no elegant solutions here. The diodes are super sensitive to current and voltage. The small batteries are necessary to maintain that balance. The cases are integrated into the lens alignment and are just too fragile to disassemble. I was tired of trying to get the laser diode out of the case intact.

While in a boring meeting at my real job, it came to me. Tape down the button and take control of the batteries with a dry contact relay. Can't fail on that path!

Setting up for success, I found out that breaking a number of these pointers was not a total loss, I became something of an authority on their anatomy. The basic laser pointer cross section is shown in Diagram 1.

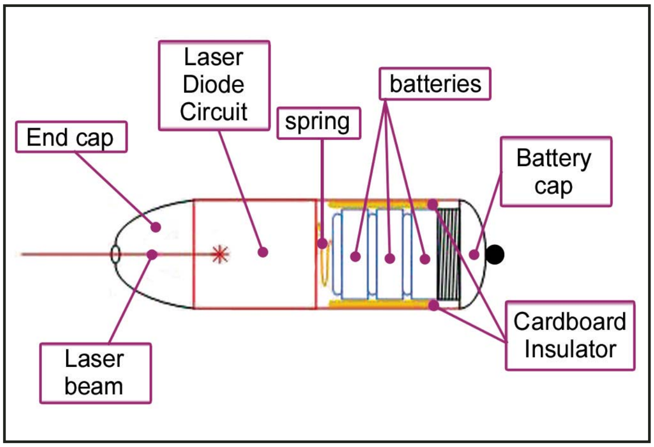


DIAGRAM 1. Laser pointer cross section.

If you study the electrical path for a second you'll notice that the current through the batteries is bridged by the end cap to the case of the pointer much like a flashlight. The individual batteries are insulated from the wall with a cardboard-like tube. To get control of the situation, we need an insulator that can be controlled by a relay. A disk-shaped insulator will sandwich between the battery cap and the back of the last battery. It will have a wire contact in the center that will be brought out of the case. We solder another wire to the battery cap to close the circuit and the laser will light up. Viola! It worked! Da 7/11 see me no more (see Diagram 2).

Some laser pointers have a flat battery cap on the inside. The one I used has an indentation. This is a necessary feature. If your pointer does not have this indentation, you can drill it out. I'd suggest a 1/4" hole about

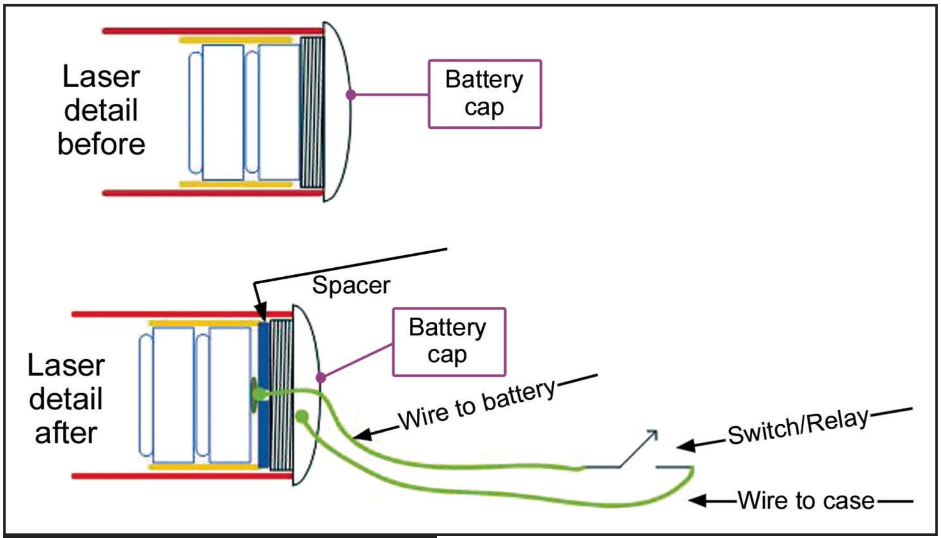


DIAGRAM 2. Laser insulator detail.


PHOTO 1. Battery cap with indentation.

1/4" deep (see Photo 1).

Step 1: *Drill the wire hole*. Remove the battery cap and decide where you want the wire hole. Slightly off center of the little key chain connection is a good place. The caps are usually made of a cheap brass so drilling is a cinch. BUT! Since you are starting the hole on the top on a rounded surface, you'll need to punch a drill guide or drill from the inside out (see Photo 2).

Use a drill press if you can. These caps are small so make sure it's in a vice. Doing it by hand will hurt (see Photo 3)!


PHOTO 2. Punch.

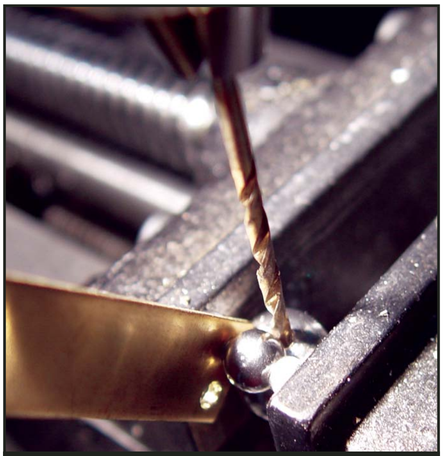


PHOTO 3. Drill.

Step 2: *Create an insulator/spacer*. I used a small clear disk cut from the plastic in one of those obnoxiously secure product packages. Cut it around the battery for shape with scissors. It does not have to be exactly round but small enough so it can go easily in and out of the pointer case. You don't want it to get stuck (see Photo 4).

Step 3: *Make a wire hole*. Punch a hole in the center of the plastic disk with an awl or pin. The hole only needs to be big enough for the stripped wire (see Photo 5).

Step 4: *Wire the disk*. Strip a piece of 20 ga wire and clamp it straight up in a vice. Slip the disk over the wire so it's sitting on top (see Photo 6).

Carefully but quickly place a bead of solder on the exposed wire. This will hold the wire in place and provide a larger contact area for the battery. After cooling, the solder and wire will look like a turnip (see Photo 7). Snip the solder ball in half with a pair of wire cutters (see Photo 8).

Step 5: *Wire the case*. Solder another piece of wire to the inside of the battery cap. It's easiest if you route it through the hole first (see Photo 9).

If any solder is hanging out of the cap, file it off so the cap has a nice flat edge to push on the insulator.

Step 6: *Heat Shrink (optional)*. Add a little piece of heat shrink tubing over

PHOTO 4. Disk and battery.

PHOTO 5. Wire hole.

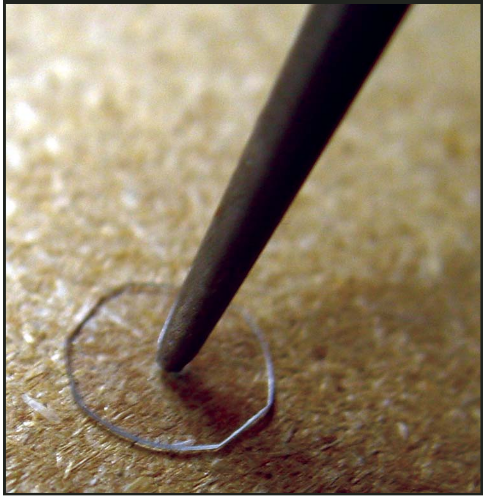


PHOTO 6. Pre-solder wire.

PHOTO 7. Solder turnip.

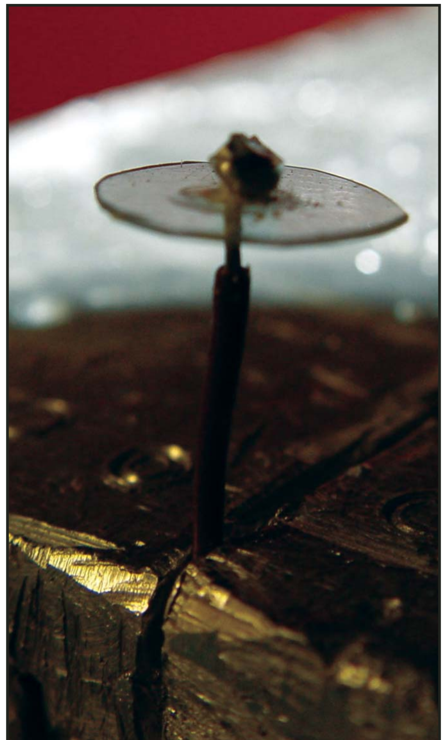


PHOTO 8. Completed contact.

the wires to protect them from any sharp edges on the wire hole.

Step 7: *Put it all together*. String the insulator wire, heat shrink, and the cap wire through the hole drilled earlier and pull the insulator wire fairly tight. It should look something like this.

PHOTO 9. Case wire.

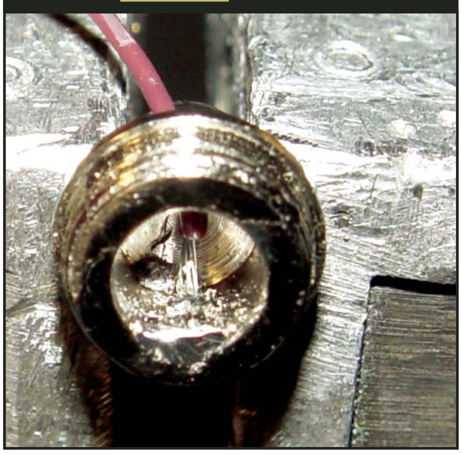


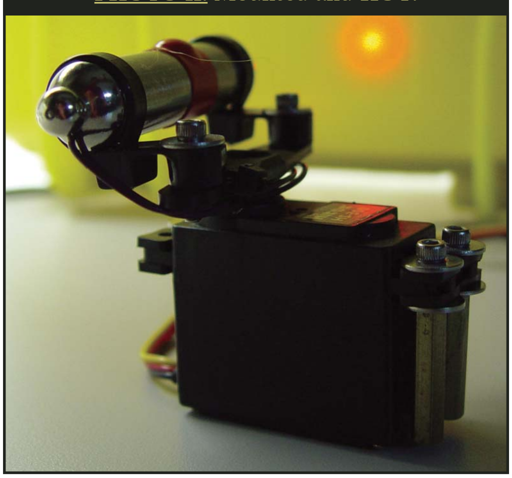
PHOTO 10. Completed construction.

Step 8: *Reassemble the laser pointer*. Screw the battery cap complete with insulator and wires back down into the case and you are done with the construction (see Photo 10).

Step 9: *Tape the button*. Wrap a piece of tape around the laser button to hold it down. You could also use heat shrink for this. With the button permanently "pushed," the ends of the wires now need to be connected to a relay or other dry contact switch to activate the laser. In general, remember the entire case of the laser pointer is electrically active. If you use a metal mounting you will need to be careful about your wiring of the relay.

Step 10: *Mount the laser on a servo*. I mounted the assembly on a long servo arm with a couple of screw-ready wire ties (see Photo 11). I wanted access to the set screw on the servo arm without cutting any ties. There are a zillion ways to do this part so be creative! Let me know how it goes. You can reach me at jim@cannibalrobotics.com **SV**

PHOTO 11. Mounted and HOT!



About the Author

Jim Miller has a Bachelor of Science degree in Physics. He has been an active roboticist since 1983 when his Apple //e was running experiments in the chemistry lab. He has written numerous articles on computer interfacing and robotics including all forms of computers. You can find out more about him and see the current state of affairs at CannibalRobotics.com

[Part 2]

INTERMEDIATE ROBOTS

Building a Laptop- or PDA-Based Robot

MEET THE 'BOTS

- HelmBot — iPaq PDA Robot (left)
- Seeker — Laptop-based Robo-Magellan Robot (below)

In last month's article, I described the hardware design of a laptop-based robot. Now, let's get into the software. The first thing you should consider is "what do I want to be able to do with this software?"

Software Requirements

The requirements for my design were:

- Must communicate reliably with an embedded microcontroller.
- Must have extensible behavior programming capabilities.
- Must have an easily customizable User Interface, to aid in troubleshooting (see Figure 1).
- Must have wireless remote operation and the ability to debug from another computer.
- Should have a Path entry for following a complex course.
- Should have Graphical mapping for navigation debugging.
- Should have a simulator, to allow testing some of the algorithms without having to actually run the robot.

Notice that the last three items are "should haves." I could still have built a successful Robo-Magellan robot without these, but they made developing a lot easier!

BY DAVE SHINSEL

High Level Design

Refer to Figure 2. The design is built around a robot behavior and control engine that performs the following:

1. Accepts user commands from either a local or a remote User Interface.

2. Processes sensory data passed to it from the PIC microcontroller or other inputs (such as a video camera).

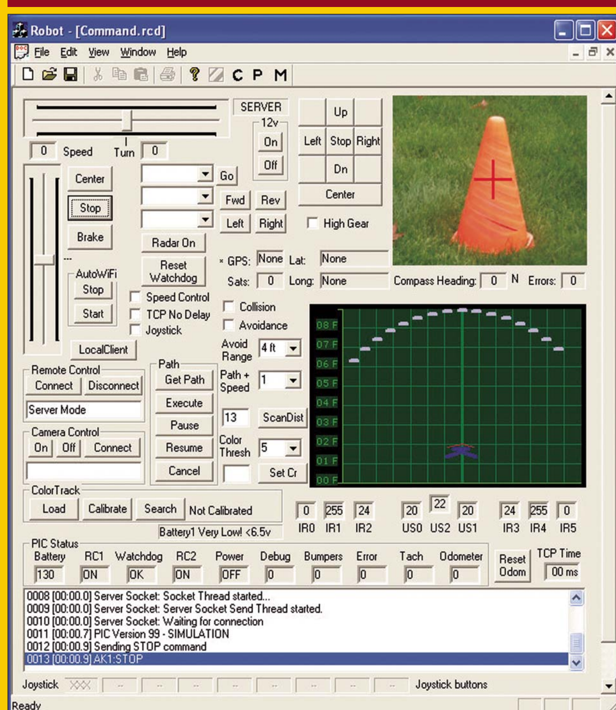
3. Based upon the requested behavior and the sensory data, determines the appropriate action, and issues commands to send to the PIC which are, in turn, sent to the hardware.

4. Sends status and debug information back to the User Interface, to aid in navigation (in the case of remote control) and debug.

A QUICK WORD ABOUT THREADS

Win32 (Windows NT and XP) provides a threading model based upon Preemptive Multitasking. This allows several tasks to all appear to run at the same time. The processor is actually letting each run for a short time-slice, and then switches to the next task. Inside a single program, these tasks are called threads. Threads are handy because they can block while waiting for some event to happen, and not cause the whole system to stop responding. Threads require almost no attention from the CPU when blocked, so performance stays high. Threads make writing complex, multitasking programs much easier.

FIGURE 1. Seeker Command GUI



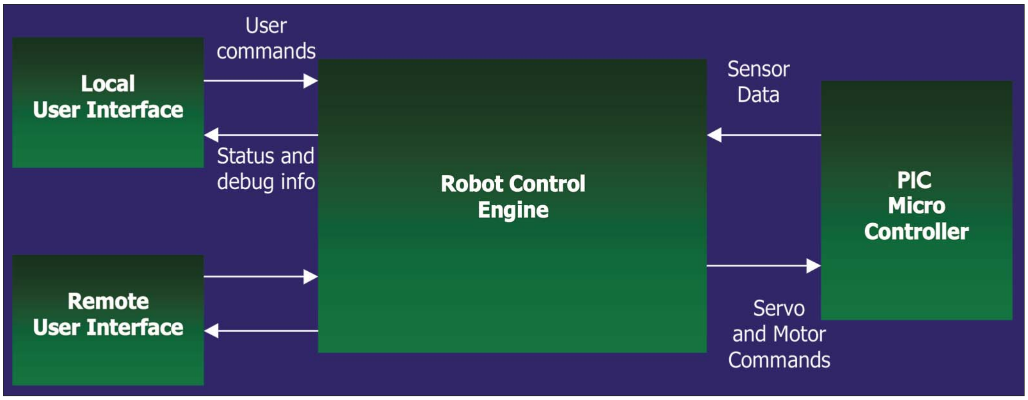


FIGURE 2. Robot Control High-level Overview

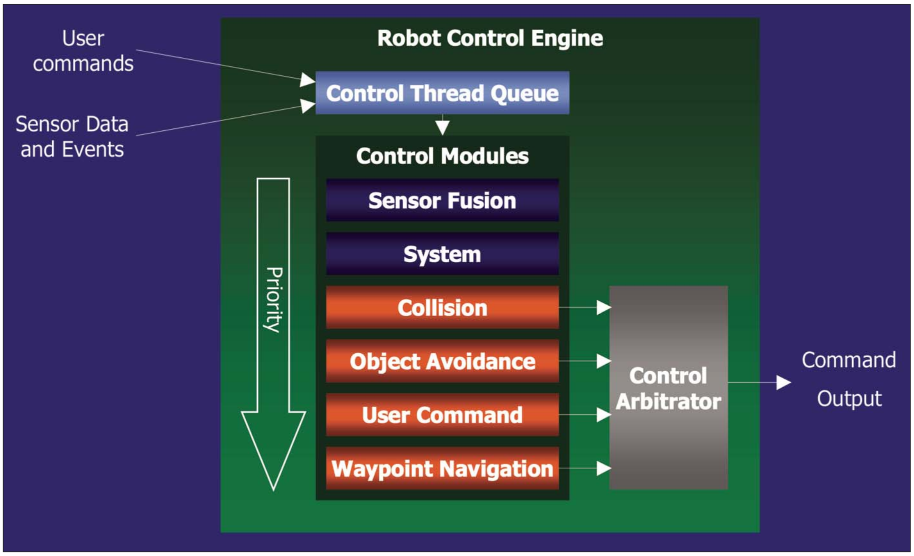
Control Engine

Refer to Figure 3. All user commands and sensory data are fed into the control engine. This engine uses a modified subsumption architecture, first proposed by Rodney Brooks at MIT. This behavioral programming model is frequently used in robots. The basic idea is this: Sensory and command data is sent to all modules and each module makes its own decisions on what the output to the motors or other devices should be. Modules are assigned relative priority, and an arbitrator blocks commands from lower-priority modules.

Let's try an example. Say the Object Avoidance Module sends a command to "turn left" while the Navigation Module sends a command to "turn right." Since the Object Avoidance Module has priority over the Navigation Module, the arbitrator will allow only the turn left command to get through.

There are times, however, when it might be desirable for a lower priority module to suppress commands from a higher priority module. For example, in the Robo-Magellan contest, robots must actually touch cones on the course to get extra points. With no way to suppress the behavior, the Object Avoidance Module would always cause the robot to swerve away from the cone! So, when the Navigation Module has locked on to a cone and is heading towards it, it will suppress the Object Avoidance and Collision modules until the cone is reached.

FIGURE 3. Robot Control Engine



Control Modules

The control engine features a pluggable architecture. New modules may be fairly easily inserted into the control engine, and the control thread will send commands and sensor data to the new module. I found the pluggable architecture so useful that I even used it for two processing modules that take care of non-behavioral tasks; the Sensor Fusion and System modules.

The highest priority module is the **Sensor Fusion Module**. This module pre-processes data that is then fed to all the other modules. It scales sensor data from raw values to a standard unit (inches), combines inputs to calculate interesting data (such as the nearest threat), and updates the robot's location on its internal map.

The **System Module** mostly tracks the health of the PIC microcontroller. It posts error messages from the PIC, reports the PIC version info, and sends "snapshots" of sensory data to the User Interface for debug purposes.

The **Collision Module** is the highest priority behavior module. The Collision Module monitors sensor data for range values that fall below specific thresholds. Examples of this are IR sensors reporting an object less than four inches away, or bumper switches being pressed. Upon triggering, the Collision Module's internal state machine takes over control of the motors and steering. It will retain control until it has completed a sequence of steps which attempt to clear the robot from the obstruction. Once the behavior has completed, control is returned to other modules.

The next highest priority is given to the **Object Avoidance Module**. It is interesting to note that this module receives exactly the same sensor data as the Collision Module, but acts differently upon the data it receives. The Object Avoidance Module keeps moving the robot forward, but will attempt to steer the robot around objects in its path.

The **User Command Module** processes all commands from the User Interface. This module allows the user to manually drive the robot, pan the camera, etc. Note that a user command to drive into a tree would be overridden by either the Object Avoidance or Collision modules. This is handy when remote controlling the robot over the Internet.

Robot **Waypoint Navigation** is the lowest priority module. When no other modules have asserted control, the Waypoint Navigation Module is free to drive the robot to its destination. If one of the other modules forces the robot off course (for example, to avoid an object), the navigation module will recalculate its route and resume heading to the next waypoint.

MORE ABOUT SUBSUMPTION ARCHITECTURE

Subsumption Architecture is much like your nervous system. Your high-level thought process may instruct your hand to pick up a pan from the stove. However, your reflex behavior upon touching the hot pan will override the initial command, because the reflex to not get burned has higher priority. If you decide you really want to pick up the pan anyway, you can suppress your natural reaction and pick up the pan. Reflex behaviors are high priority; they get processed even when your mind is on something else. High-level thinking takes longer to process stimulus data and decide what to do. But, once a decision has been made, the lower priority behavior can temporarily suppress reflex behaviors. A good example of this is when you consciously decide to hold your breath.

Internal Map

The robot software utilizes an internal map based upon a grid that is just over one mile square. All coordinates are expressed in inches. Since a 16-bit word can hold a number up to 65,535, a single 32-bit value can hold the X and Y coordinates for any location within a 1.03 square mile area.

The map Origin (coordinate 0,0) is in the South West corner of the map. The map may be anchored to the real world by associating any one point on the map with a GPS coordinate. The Origin is automatically calculated as an offset from that point. It is interesting to note, however, that the map is only required to be anchored to real world coordinates if GPS is being used. For the SRS Robo-Magellan contest in Seattle, I did not use GPS at all. Instead, I chose an arbitrary start location set to (1000, 1000). All real world features — such as cone placements and obstacles — are calculated automatically as offsets from that point. The only critical requirement is that all interesting information must stay within the one square mile boundary of (0,0) to (65,535, 65,535).

Navigation

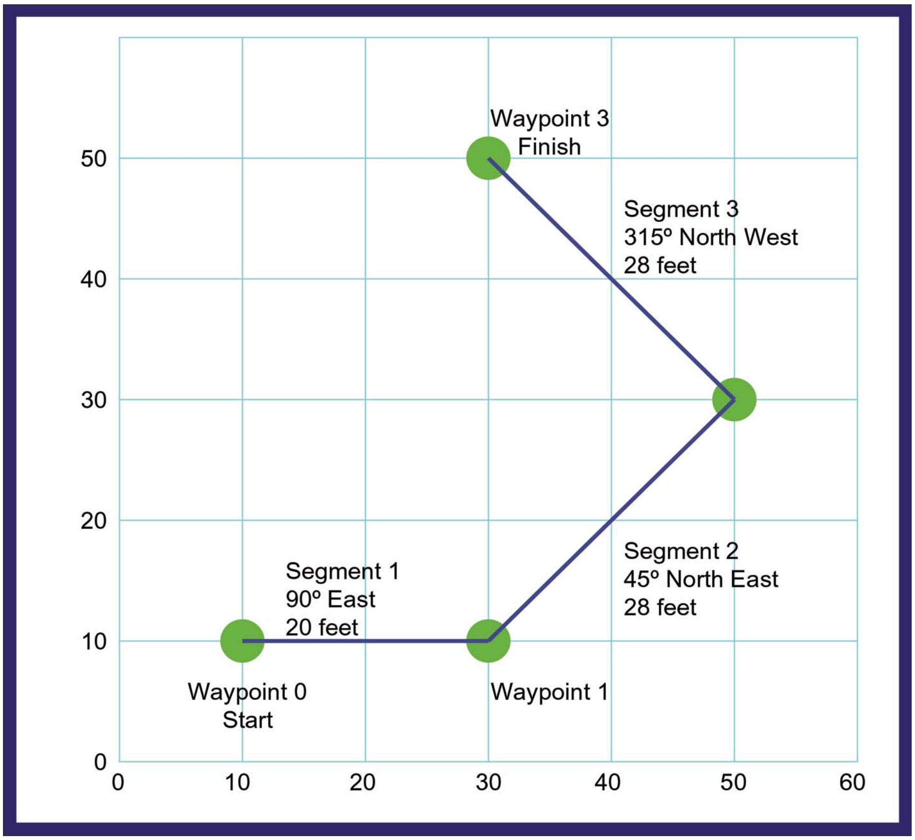
There are many types of navigation that might be interesting to a robot builder. For the purposes of this article, I will focus on pre-planned navigation. In pre-planned navigation, the robot is provided with a starting location, final destination, and *Waypoints* along the way.

Using an internal map, the robot must be able to find its way from one Waypoint to the next, avoiding obstacles along the way. I use the term *Segment* to refer to the connecting lines between Waypoints, and the term Path to represent the full collection of Waypoints connected by Segments (see Figure 4).

The Path Entry Dialog (see Figure 5) is used to enter the various Segments the robot is expected to follow. First, the Start Waypoint is entered in absolute coordinates. By default, this is (1000, 1000). Next, absolute direction (in degrees from North) and distance (in feet and inches) is entered for each segment. Once all of the segments are entered, pressing the "ReCalculate All" button will calculate the absolute location of all the other Waypoints.

Waypoints are associated with an (X,Y) location coordinate. In addition, a Waypoint may optionally have a

FIGURE 4. Internal Map — Segments and Waypoints



ATTRIBUTES

Waypoint Attributes:
- Waypoint location (X,Y)
- Landmark Type
  - None
  - Cone
  - Drop Off
  - Pole
  - Tree
  - Wall
- Landmark Range (in inches)
- Landmark Direction (in degrees)

Segment Attributes:
- From and To Waypoints
- Distance (length of segment)
- Direction
- Speed
- Behavior
  - Compass
  - Compass+GPS
  - Follow Wall
  - Follow Dropoff
  - Follow Hall
  - Find Doorway
- Optional: Follow Wall parameters
  - Left/Right
  - Distance

Landmark Type associated with it. If the Landmark Type is set to "Cone," the robot camera will actively search for a cone while heading for the landmark. If the Landmark Type is "Pole" or "Tree," the robot will search for a narrow object in the right location, and head for it. When the robot reaches the specified distance from the landmark, it will consider the Waypoint reached, and head for the next Waypoint. Landmarks are very useful for correcting errors that build up as the robot navigates along the path.

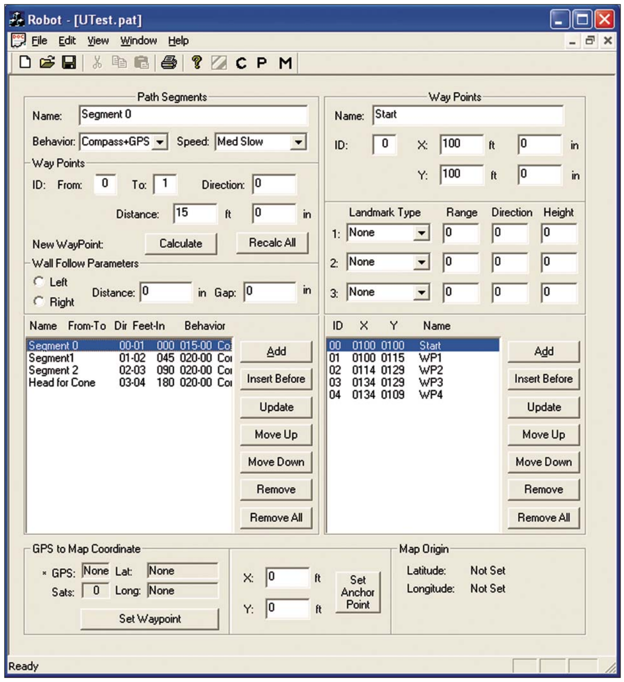


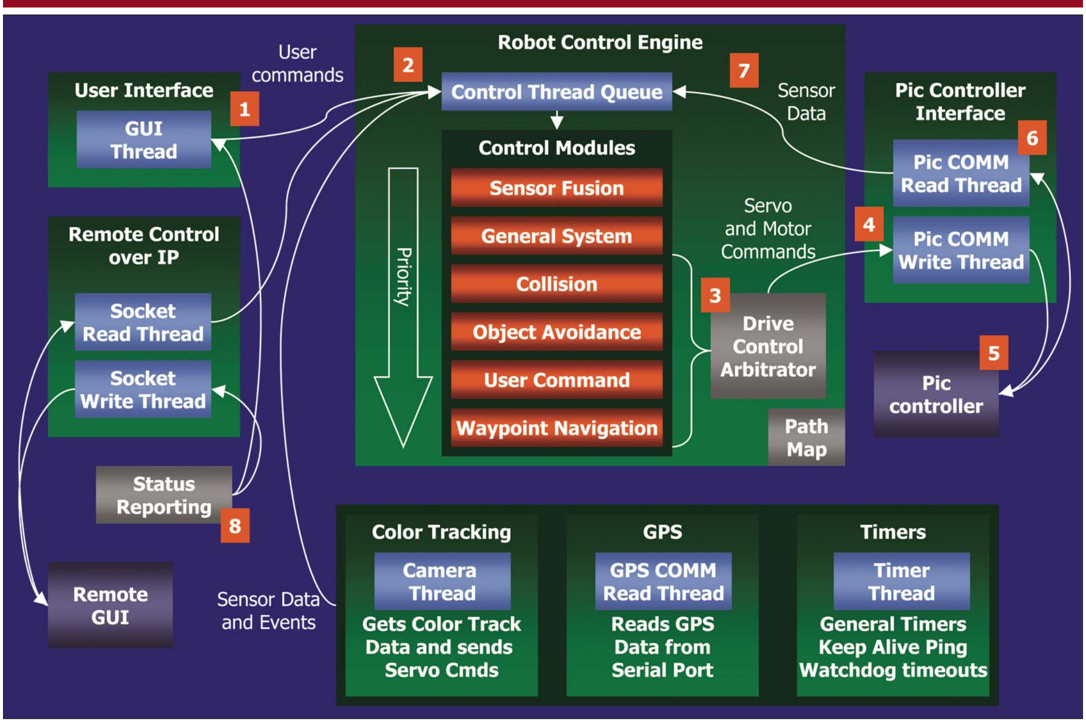
FIGURE 5. Path Entry Dialog

Segments can have attributes as well, which help the robot successfully navigate specific portions of the course. For example, if the Segment Behavior is "Follow Wall," the robot will drive a path parallel to the wall. Additional parameters are provided to indicate left or right side and distance to maintain from the wall. A behavior of "Follow Hall" or "Enter Doorway" will cause the robot to search for an open area between two objects, and head for that middle area between them.

Simulator

The Robot Control Software has a built-in simulator. The program is available for download from the *SERVO* website (**www.servomagazine.com**). This will give you a good feel for the robot's capabilities prior to jumping into the source code. Once you download the program, simply create a new Path and enter some data. Next, create a new Map. If you tell the robot to move, the robot icon on the map will move in response to your commands. If you tell the robot to follow the path, it will to the best of its ability. Try it!

FIGURE 6. Control Software Data Flow



Connecting it All Together

Refer to Figure 6.

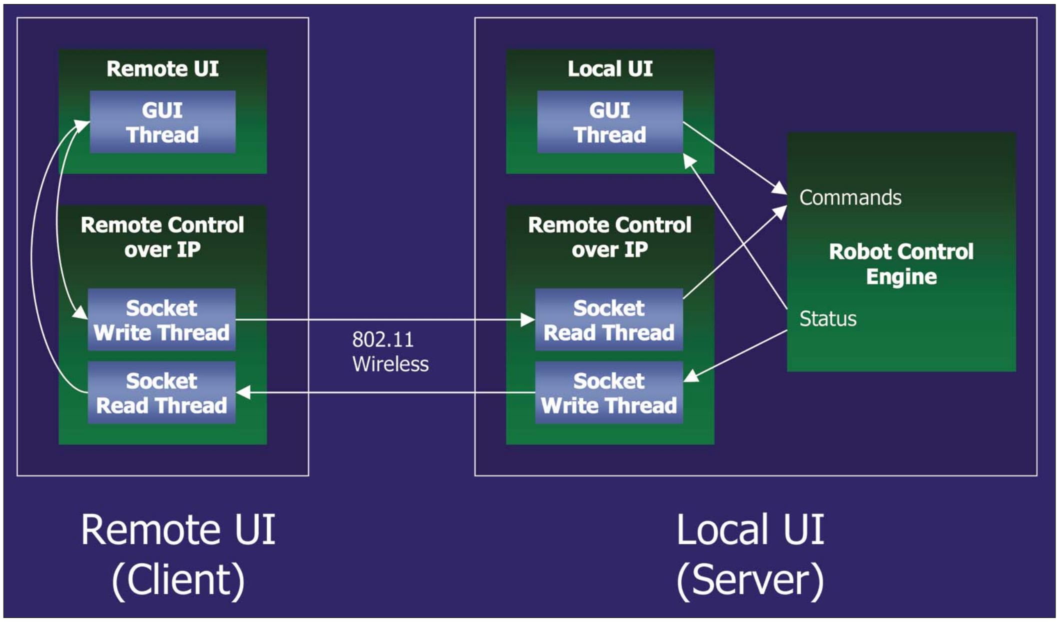


FIGURE 7. Remote User Interface

This detailed block diagram of the full system shows how all the parts of the system interact with each other. First, notice the nine threads indicated by blue boxes. Each of these threads block while waiting for some event to happen. Most are tied to a "command queue." When one thread wants to send data to another thread, it puts the data in the other thread's command queue.

Let's walk through the model. Assume the user presses a button to increase the robot speed. The User Interface GUI thread (1) will get the User Event (button pressed), and place a command into the Control Thread Queue (2). The Control Thread will pull the "set speed" command from the queue, and pass it to each of the Control Modules. The User Command Module will act on the command and issue a change speed command to the Drive Control Arbitrator (3). Assuming no other module has issued a higher priority command, the Arbitrator will send the command to the serial queue in the PIC Comm Write Thread. The PIC Comm Write Thread (4) will send the command to the PIC controller via RS-232. The PIC controller (5) will act on the command, and adjust the motor speed.

Now, let's assume an object has been detected ahead by one of the sensors. The PIC controller (5) will send a status packet to the host computer. The PIC Comm Read Thread (6) will read the status packet from the serial port, and post it to the Control Thread (7). The status is sent to each of the Control modules in turn. If the object is close enough, the Collision or Object Avoidance module may send a new command (perhaps Stop or Turn), to the Drive Control Arbitrator, and the cycle repeats.

There is a global Status Reporting module (8) that is accessible to all threads. Debug messages, sensor status, and other useful information are sent to the GUI thread by the Status Reporting module.

In addition to communication from the PIC microcontroller, there are Camera, GPS, and Timer modules that also send their data to the Control Engine. The interfaces to these will be discussed in Part 3 of this series.

Remote Control and Monitoring

One nice feature of this software is that the robot can be controlled and monitored from a remote location, such as over the Internet or from a local PC using 802.11 wireless. When the Robot Control Software is compiled, there is a "build switch" that allows either the Server or Client version to be built. Both the client and

server versions are built from the same source, eliminating the need to maintain two separate projects.

Figure 7 shows how the Client (remote PC) and the Server (the laptop on the robot) communicate. Standard Windows WinSock is used to create sockets on both machines, and threads are created to read and write to the opposite machine's socket. When a remote (client) is connected to the robot, all the status and debug information is sent to the client, and all commands from the Client UI are sent to the robot.

For remote control and "remote presence," audio and video can also be sent. Currently, Microsoft Netmeeting is used to handle the audio and video, but other mechanisms can be used, as well.

For detailed debugging (stepping through code), I often use the Microsoft Remote Desktop. This is a great feature of Windows XP that allows you to take control of a computer from a remote location. However, it's not very good during actual robot operation, because the overhead can slow the robot responses down too much. During robot operation, I rely more on the remote log information that I receive on the client to understand what the robot is doing, and why.

ABOUT THE AUTHOR

Dave Shinsel has been a hardware and software engineer for a number of companies including Hughes Aircraft, Epson Printers, Mentor Graphics, and for the last 12 years, Intel Corporation. At Intel, Dave manages a software engineering team for the Consumer Electronics Group in Portland, OR.

Conclusion

Hopefully, this article has provided you with a good overview of the design approach I have taken for my robots. As mentioned earlier, a binary version of the Robot Control Software is available for download from the *SERVO* website, so you can get a good feel for the robot's capabilities.

The source code to all of the software discussed in this article is available from my website at **www.shinsel.com/robots**

To compile the software without modification, you will need Microsoft Visual C 6.0 (MSVC6). I believe you can also use Microsoft .Net, but I have not tried it yet. And, of course, since it is source code, you can convert to Linux, or whatever you want!

Next Month

In Part 3 of this series, I will go into detail about the PIC microcontroller software, how the Laptop and PIC communicate, GPS, and color tracking with a USB camera. Until then, the thought for this month is: "Real programmers don't use comments. If the code was hard to write, it should be hard to understand." **SV**

Serial Communication Without A Serial Port

Bit Banging Fundamentals

BY PAUL KAFIG

IN THE JANUARY 2005 ISSUE OF *SERVO*, JACK BUFFINGTON SHARED A WELL-WRITTEN AND INFORMATIVE ARTICLE THAT INSTRUCTED THE READER HOW TO CONNECT AN LCD DISPLAY (WHICH USES THE POPULAR HD44780 DRIVER) TO A PROCESSOR. WHILE THIS DISPLAY HAS FEATURES ABOVE AND BEYOND SIMPLE TEXT DISPLAY (WHICH MR. BUFFINGTON MENTIONS IN HIS ARTICLE), I COULDN'T HELP THINKING THAT THERE ARE SIMPLER, FASTER METHODS TO ATTACH AN LCD TO A PROCESSOR OR ROBOT. THE MOST OBVIOUS SOLUTION IS TO USE A SERIAL CONNECTION TO A SERIAL LCD (AS OPPOSED TO THE PARALLEL CONNECTION OF THE HD44780). THIS WOULD NOT ONLY BE EASIER, BUT ALSO USE FEWER PINS (ONE PIN FOR SERIAL OUTPUT VERSUS AT LEAST EIGHT FOR PARALLEL OUTPUT; ONE COULD ALWAYS HARDWIRE THE CONTROL LINES TO VCC OR GND).

STILL, THE PROBLEM MANY OF US FACE IS THAT OUR PROCESSORS ONLY HAVE ONE SERIAL PORT, AND WE NEED THAT FOR DIAGNOSTICS. Or worse yet, our processors don't have any serial ports. The solution here is to use a technique known as "bit banging."

Bit banging provides a useful way for your processor to communicate serially with the outside world without having the need for a serial port. This article will cover the theory behind bit banging and also demonstrate an example by sending a string of characters to the serial port of a PC, which can be seen by a terminal program such as Hyperterminal or Procomm.

The PIC16F84 is the processor of choice in this project along with the C programming language. If you purchased a PICSTART Plus development kit (the one typically used by PIC hobbyists), you received a programmer, PIC16F84 processor, and the PICC LITE C compiler specifically for this processor. This development kit is available from Microchip Technology (**www.microchip.com**) as part number DV003001. If you have a different kind of processor, don't worry. The concepts presented in this article can be implemented in many different ways. More on this later.

Theory

Figure 1 shows the format of a byte (eight bits) of data as it is sent out of a processor's serial port. Voltage levels here are typically five volts (high) and zero volts (low). When the processor is not sending data, it idles in the high state. To communicate, the processor initially brings the line low (the start bit), sends each bit (least significant bit first), and then brings the line high once more (the stop bit). Since there is no clock signal, the *timing* of these bits is critical.

How much time elapses between each bit transition is what determines the baud rate. Our goal here is to wiggle the logic level of a processor's general purpose I/O pin in such a way as to mimic the operation of a serial port's output. This is what bit banging is all about. For more detailed information about serial communications and the RS-232 protocol, type "RS-232 tutorial" into any Internet search engine and you will find an abundance of information.

Procedure

In this project, we're going to build a test fixture to send out a serial string at 1200 baud. We'll follow a three step process:

- Determine the length of each bit in the byte.
- Build the hardware (text fixture).
- Write the software.

Determining the Length of Each Bit

The easiest way to determine this information is to look in a book or standard. "Serial PIC'n"[1] has one such table. But if you don't

FIGURE 1

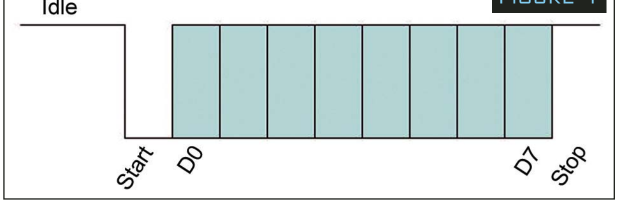
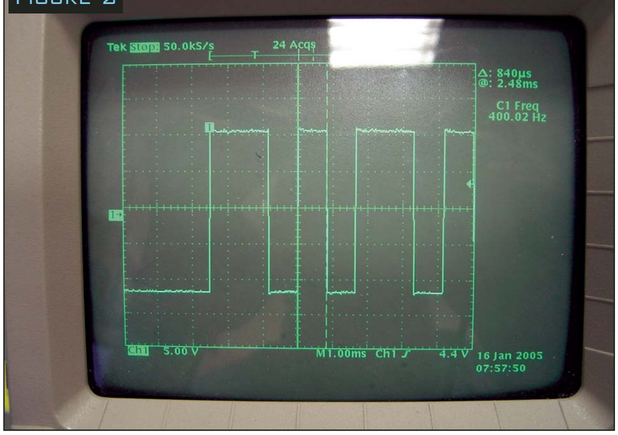


FIGURE 2



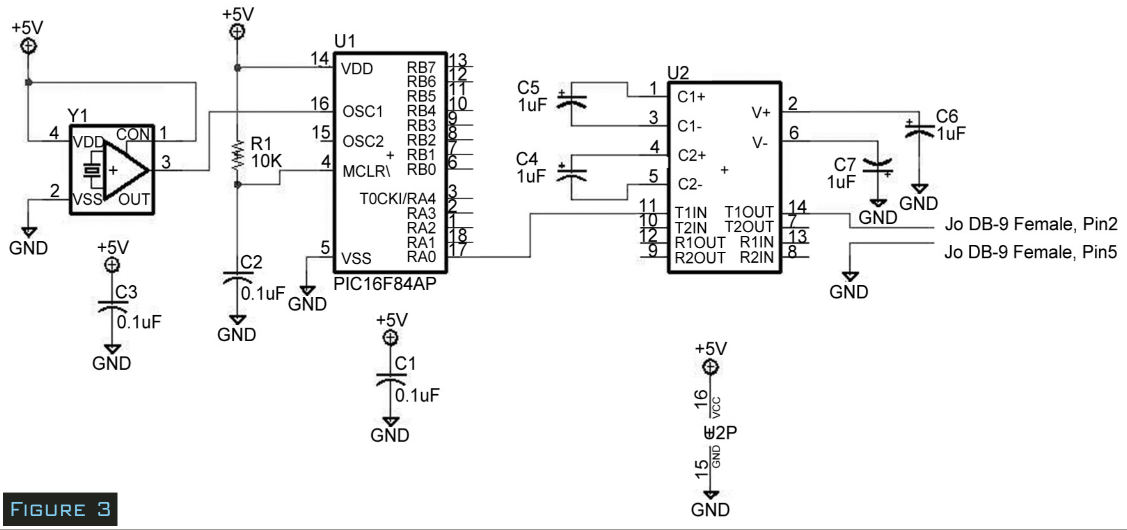


Figure 3

Reference	Description	Maufacturer	Part Number
C1-C3	Capacitor, 0.1 µf, Ceramic	BC Components	K104K15X7RF5TH5 or =
C4-C7	Capacitor, 1.0 µf, Tantalum	AVX Corporation	TAP105K020SCS or =
R1	Resistor, 10K, 1/4 watt, 5%	Yageo	CFR-25JB-680K or =
N/A	DB-9 Female Connector, Solder Cup	AMP/Tyco	747905-2 or =
Y1	Xtal Oscillator, 4.0000 MHz	Epson	SG-531P 4.000MC or =
U1	PIC16F84 Microcontroller	Microchip Technology	PIC16F84-04I/P or =
U2	IC, Dual EIA-232 DRVR/RCVR	Maxim	MAX232N or =

Figure 4

TECH TIDBIT

RS-232 communication is sometimes called *asynchronous* (without clock). There are also *synchronous* serial protocols, such as I²C.

want to take someone else's word for it (or just like to find the answers for yourself), there's a little trick you can play. By connecting an oscilloscope between the transmit and ground pins of your computer's serial port (pins 3 and 5 on a DB-9 male) and continuously sending a capital "J" or "R" through the serial port (by way of your computer's terminal program), you can identify the bits in the ASCII byte.

I like letters like "J" and "R" because they have alternating 1s and 0s in their bit stream and that makes it easy to pick out individual bits. Identify an individual bit and measure its width. To produce the waveform shown in Figure 2, I sent the letter "J" to the serial port. The width of the bit in the center (between the cursors) is about 840 µs.

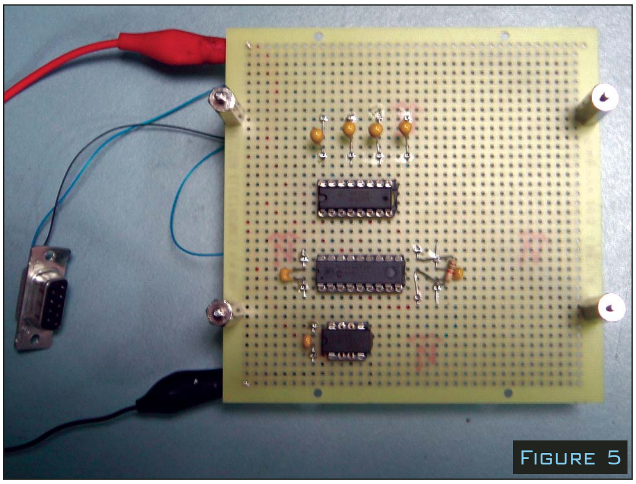


Figure 5

The Hardware

Figures 3 through 5 show the schematic of the hardware used in this project, a bill of material, and a photo of the actual hardware. Starting from the left side of the schematic, Y1 is the resonator for the PIC. Although the PIC gives several options for driving its internal clock, I like resonators for two reasons.

First, they're only one part as opposed to a crystal plus two capacitors, and second, they're pre-tuned to oscillate precisely at the desired frequency. In the center we see U1, which is the PIC. A simple RC network is used for the master reset. And finally, at the right, is the MAX232 driver to convert the 0 to 5 volt signal from the PIC into the ±12V (approximately) signal required for RS-232 communication to the PC. C1 and C3 are bypass capacitors for high frequency transients and should be located as close as possible to their respective ICs. Notice that a particular power supply is not indicated. The circuit only requires five volts. This can be done with a benchtop power supply or an additional IC for regulation. The choice is yours.

One important note here: the MAX232 driver also does one other thing. It inverts the signal from the PIC. The sketch in Figure 1 is correct for the asynchronous signal coming from a UART, PIC, or other RS-232 signal generator. The signal is inverted when it gets transmitted over an RS-232 cable. So if you're connecting your bit banging output to a serial display, make sure to read the data sheet carefully and be sure to understand what signal levels and polarity the display is expecting. Otherwise, the display may be damaged. If you're

transmitting to a computer's serial port, it's a good idea to run the PIC's signal through a chip or other circuit that will give the correct levels and polarity.

As for constructing the test fixture, I wire wrapped mine. For this kind of project, soldering is just too much trouble and using a breadboard is too unreliable. It took me less than an hour to construct the hardware for this project.

The Software

For the software, we're going to write two programs. The first will establish the time delay between each bit transition. The second will perform the actual bit banging.

Figure 6 shows one possible version of the first program. This program simply alternates between high and low logic levels (equivalent to a *really* long character consisting of alternating 1s and 0s). If you're using an oscilloscope, you can look at the signal and measure each bit's width. If you don't have a fancy oscilloscope, but still have some rudimentary way to measure frequency (many reasonably priced multimeters have this function), you can still tune your PIC's delay function. Since we know that the width of one bit is around 840 µs, a full high to low to high period will be twice this, or about 1,680 µs. Inverting this value will give you the frequency, about 600 Hz. Connect your probes between the bit banging pin (pin 17 in this case) and ground and try to attain this frequency.

Figure 7 shows one possible version for the final test code. Starting from the main() statement, I first define the test string and a variable to keep track of the character within the string that I'm printing (the index). The while loop cycles through each character of the test string. Looking at how the bit banging function works, RA0 is first brought low for one bit

```
#include <pic.h>
__CONFIG(0x3FF9);                          // configuration bits
                                           // Oscillator XT
                                           // Watchdog timer OFF
                                           // Powerup timer OFF
                                           // Code protect OFF


void delay(void);

main(){

        TRISA = 0x00;                      // port A is an output port

        while(1){
                RA0 = 1;
                delay();
                RA0 = 0;
                delay();
        }

}

//-----------------------------------
//            Functions
//-----------------------------------



//        This function gives an 840us delay
//        with a 4MHz oscillator
void delay(void){
         int i;

         for (i=0; i<47; i++);
}
```

FIGURE 6

FIGURE 7

```
#include <pic.h>
__CONFIG(0x3FF9);                          // configuration bits
                                           // Oscillator XT
                                           // Watchdog timer OFF
                                           // Powerup timer OFF
                                           // Code protect OFF

//-----------------------------------
//        Function Prototypes
//-----------------------------------
void delay(void);
void bit_bang(char ch);

//-----------------------------------
//          main program
//-----------------------------------
main(){
       char string[] = "Servo!\r\n";                    // our test string
       char index;

       TRISA = 0x00;                            // port A is an output port

       while(1){                                        // infinite loop
              for (index=0; index<9; index++){          // print the test string...
                              bit_bang(string[index]); //   one character at a time
              }
       }
}

////////////////////////////////////////
//
//      delay
//
//      This function gives a 840us delay
```

continued ...

Figure 7 continued

```
//      with a 4MHz oscillator
//////////////////////////////////////////
void delay(void){
        int i;

        for (i=0; i<47; i++);
}

//////////////////////////////////////////
//      bit_bang
//////////////////////////////////////////
void bit_bang(char ch){
        char cnt;

        RA0 = 0;                                          // start bit
        delay();

        for (cnt=0; cnt<8; cnt++){
                RA0 = (ch >> cnt) & 0x01;                 // bit 0 to 7
                delay();
        }

        RA0 = 1;                                          // stop bit
        delay();
}
```

period. This is the start bit to tell the receiving hardware that a character is about to be sent. Next is a loop that counts from 0 to 7, inclusive. Each time through the loop, the bit pattern of the character is shifted right a number of times equal to the loop iteration. This shifted value is logical ANDed with 1. This operation just masks out every bit except the right most (least significant) bit. Whatever this right most bit is, that's the value that gets placed on the pin.

Finally, the pin goes high for one bit period to signify the stop bit and the end of the character. The output of this program is shown in Figure 8.

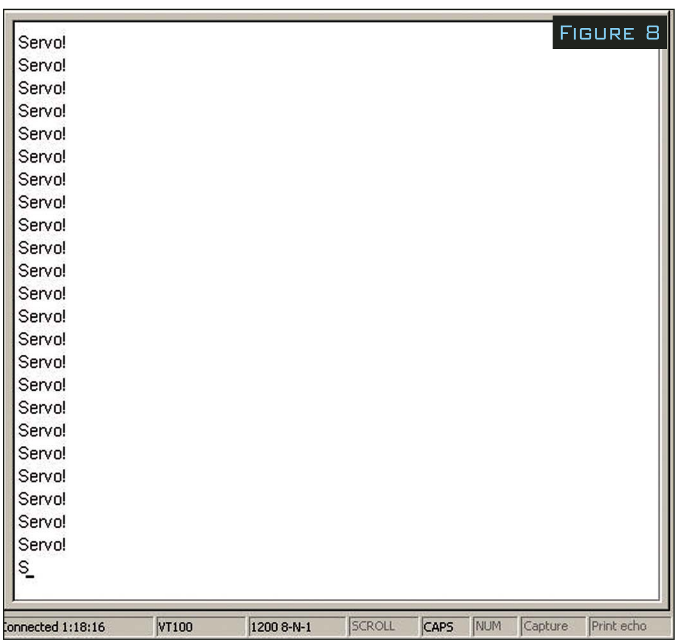
Conclusion and Ideas for Further Experimentation

Although this article focused on using bit banging to send a series of characters to your computer's serial port, the concepts here are easily adaptable for other applications. For example, if your goal is to add a display to your robot, simply use the output from the PIC without the level conversion and attach this to a serial display. An excellent source for serial displays is Scott Edwards Electronics, Inc. (**www.seetron.com**).

Also, if the PIC is not your processor of choice, that's not a problem. These bit banging concepts can be applied to any processor that is fast enough.

Here are some other exercises and/or ideas to try:

Figure 8



• If you're using a MAX232 driver (or equivalent circuitry) to level shift the serial output from your processor, you can add a second "serial port" to the project by bit banging out a second pin. See if you can modify the code to use the same C function for both pins (I suggest you pass a variable to the function).

• You can bit bang a message to a serial servo controller and use this method to control the servos in your robot.

• Write the code to *accept* a serial signal from a computer. In this case, you would have to detect the start bit and then wait the appropriate amount of time before sampling the data on the pin. You would then need to concatenate (and possibly reverse, depending on your algorithm) the bits before processing the incoming character.

• Do a little research to determine how parity is calculated and add a parity bit.

• Change the baud rate of the bit banging in this project. Just remember that the higher the baud rate, the less forgiving the timing will be and the more accurate your processor will have to be.

• Processor too slow? Bit banging can be done *totally in hardware*. Use your favorite processor to write a byte (character) to a latch. When your processor gives the go-ahead, the bit banging can be done in hardware. You can use discrete logic ICs or program an extra port (or many ports) into an FPGA or CPLD.

Have fun bit banging away! SV

Resource

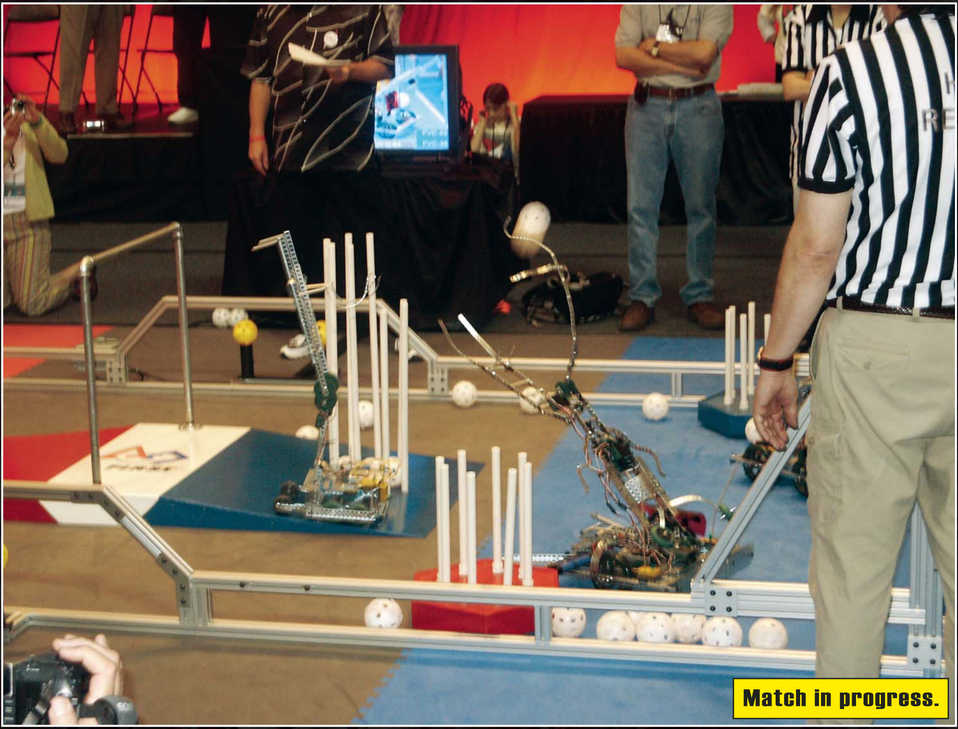
[1] Stevens, Roger L. Serial PIC'n, Square 1 Electronics, Kelseyville, CA, 1999.

2005 VEX CHALLENGE

by Lester "Ringo" Davis

Many readers of *SERVO* know about the FIRST competition that is held every year. There have traditionally been two different divisions — the FIRST LEGO League (FLL) and the FIRST Robotics Competition (FRC). The FLL is designed for middle schoolers and FRC for high schoolers. The problem is that it is a huge leap from one group to the other. The FRC robots can be eight feet tall or more when fully extended, weigh well over 100 pounds, and cost several thousands of dollars to build. So, this year FIRST decided to try something new: the VEX challenge.

For this challenge, participants received four VEX kits that each consisted of a radio transmitter and receiver, three motors, one servo, eight wheels, assorted gears, a microcontroller, and more nuts, bolts, and metal mechanical parts than you could count. These VEX kits are now available at local RadioShack stores and were initially developed for FIRST. I was lucky enough to find out about the challenge and was selected to participate as a mentor for my 15-year-old son and his two friends who made up VEX Challenge team #8.


Match in progress.

The Challenge

Like FRC and FLL, we had no idea what the challenge would be or what would be in the kit before the kickoff day. Everyone receives the same information on the same day so that no one has a time advantage. The challenge this year was almost exactly the same as the FRC challenge from last year, except it was scaled down. Each team has a human player that must throw wiffle balls into a mobile goal or a stationary goal. The robot must herd balls to the human player in order for the human to throw them into the goal. There is also a large rubber ball that caps the mobile goal at the beginning of every round. If the robot can remove that ball, then replace it after balls are in the goal, then the score is doubled. There is also a large bonus if the robot can hang from a chin-up type bar in the middle of the arena.

Figure 1 shows the arena before a match starts. There are two teams on the red end and two teams on the blue end. The teams are randomly selected and must work together. Your ally in one round may be your

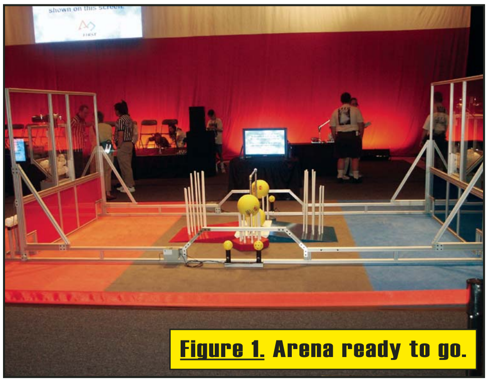


Figure 1. Arena ready to go.

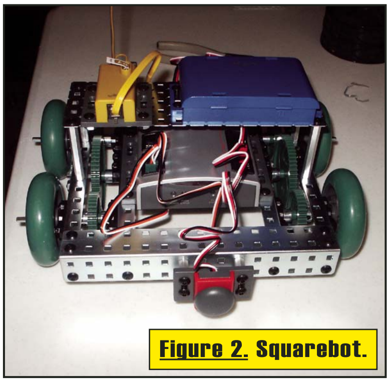


Figure 2. Squarebot.

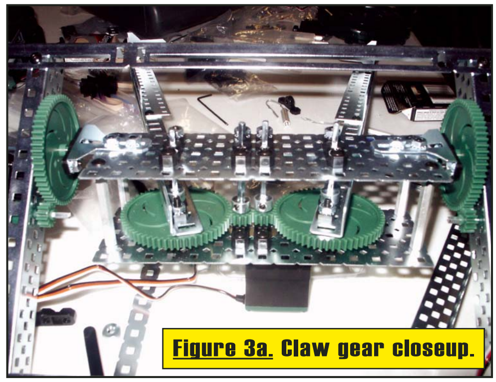


Figure 3a. Claw gear closeup.

competition in another. With each round, the teams change so the strategy changes depending on the capabilities of each robot. Some teams had simple bots that only corralled balls, some had arms for picking up the large rubber balls, and some had arms for hanging from the central bar.

Building the Robot

The kit for the challenge included four complete VEX starter kits. We were only allowed to use the total number of parts from three kits, so we had an entire kit as a spare. After looking at what we had to work with, we decided to start out trying to build a bot that could pick up the large rubber ball from the ground or from the mobile goal, and put it back on the mobile goal. When that was completed, the plan was to start work on the bar-hanging mechanism.

The first night the team got together, they started going through the manual included with the kit. It stepped them through building a simple robot called Squarebot. It explained how to use the motors and gears to build a four wheel drive geartrain, how to use the radio, and how to wire up sensors to the microcontroller. By the end of the first night, the team had Squarebot running around under radio control. It was equipped with bump sensors that disabled it for a few seconds if either one was hit. It was designed as a soccer playing bot, but the boys just used it for fun. Figure 2 shows the completed Squarebot.

Team #8 now had a good idea on how the motors and radio worked and were ready to start building. We first tackled the claw that would pick up the ball from the ground and place it on top of a 15-inch goal. We used a motor and a few gears to build a mechanism that would open and close two long arms. After that, we added a frame around that portion and two motors that could tilt the arm up and down. The long arm needed lots of torque to lift the ball, so we used two motors on opposite sides of the frame. An interesting aspect of the VEX controller is that you can plug in motors in different configurations so that they work together, such as in two motors on one side for four-wheel drive, or so that they spin in opposite directions as in our configuration. Figure 3a shows a close-up of the gears and 3b shows a better view of how the motors are mounted.

Once the claw was assembled, we built a frame to hold it up off the floor to test it. Figure 4 shows the claw successfully lifting the rubber ball into the air. If you look closely, you will see an extra gear in the lifting mechanism between Figures 3 and 4. On our first lifting test, the motors were not strong enough to lift the ball, so we added a second stage. Each stage reduces the gear ratio by 5 to 1, so the total reduction is 25 to 1. With this configuration, we had plenty of power to lift the ball.

So now we had a working lifter and just needed to make it mobile. The team spent a couple nights building a four-wheel drive unit and bolting it onto the claw.

When we tried to pick up a ball with the new bot, it promptly tilted over on its face. A couple wheelie bars in the front of the robot fixed this problem and also served as a nice ball corral. Figure 5 shows the completed robot ready to go.

Practice

The first day the kits came out,

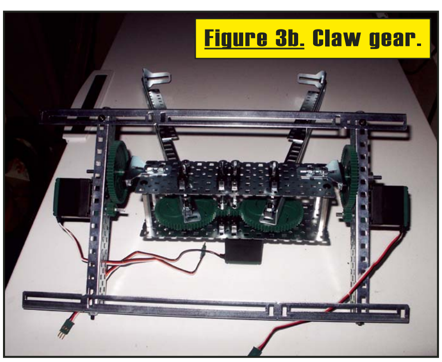


Figure 3b. Claw gear.

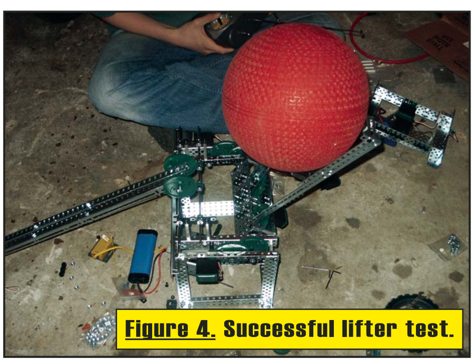


Figure 4. Successful lifter test.

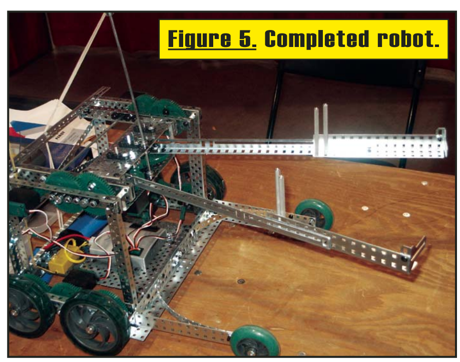


Figure 5. Completed robot.

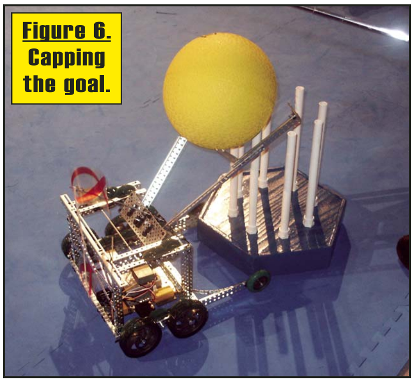


Figure 6. Capping the goal.

Figure 7. Alliance #8.

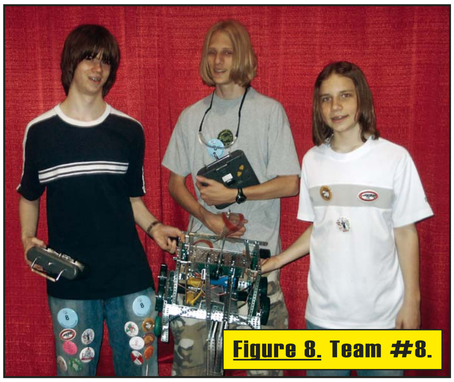


Figure 8. Team #8.

everyone knew about each other's team. I set up a Yahoo! group and emailed all the local teams about it so that we could communicate and help each other out. This turned out to be useful as one of the other teams brought up the idea of having a practice scrimmage a week before the competition. One of the teams secured a church gymnasium for us to use one Saturday afternoon. In order to practice for the event, each team was given the exact dimensions of every part of the arena so that replica parts could be made. Some people only made the goals while other people built almost the entire arena. During the scrimmage, everyone brought the parts they had, so it allowed us to practice against each other and for everyone to get a good idea of what the real thing would be like. Figure 6 shows our bot putting the rubber ball on top of the mobile goal.

It was a good thing we were there because we found a problem with our design. While we had plenty of power from the motors now, the inside of the gears were stripping out. The gears have a square hole in the center that goes on a square shaft that also feeds into the motor. The square holes had turned round under the stress of picking up the ball with the long arms. We thought about doubling up the gears, but that would mean a total redesign and rebuild of the robot.

The kit includes small plastic bearings with metal inserts and square holes that we had already used to mount the axle to the rotating part of the arm. We took some new gears and bolted these bearings to the gears to strengthen them and did not have further problems.

The Competition

When we arrived at the Georgia Dome, we proceeded to the pits to set up and start practicing. We started looking around at all the other designs and were amazed at how different everyone's robots were. There was a day and a half of qualification rounds followed by the finals on the afternoon of the second day. There was a very large screen set up in the pits showing everyone's rankings as the day progressed. Our design used two transmitters and two receivers. This allowed one person to drive, and the other to work the arm.

During our first round, we had placed the receiver crystals in the wrong receivers so our bot looked like it was having a fit. Needless to say, we did horribly and lost the match. We noticed after that we were ranked 51 out of 52 teams. Luckily for us, there were lots of qualification rounds. We did better after that and won a few rounds and lost a few rounds. At one time, we made it up as far as 24. We were happy just to make it into the top 50%.

After all the qualification rounds were finished, the top eight teams picked their alliance partners for the final rounds. Each alliance consisted of three teams. Of course, the first place team picked the second place team as an alliance, along with one other team. Everyone thought they would be unbeatable. Now out of choices, the eighth place team picked us to be in their alliance. We were elated, until we found out that we were matched up against the first alliance, which meant the top two teams, both of which were undefeated.

The quarter finals were two out of three matches. We lost the first match but came back to win the next two, sending us to the semi-finals and knocking out the undefeated teams! During the semi-finals, one of our teammates stripped a gear, so we were frantically looking around for someone that would lend us some spare parts. A girl from another team saw our distress and quickly ran over to her team's supplies and gave us what we needed. It wasn't until the next match started that we realized that she was from the competition. I don't remember the team name, but VEX team #35 really showed what "gracious professionalism" is all about.

Our teammate had about two minutes to rebuild his lifting mechanism with the new parts and was just able to get it finished in time. We went on to the semi-finals in four matches (one was a tie) and made it to the finals. We ended up losing the finals in the third match to an alliance that included teams sponsored by RadioShack and NASA. If you are going to lose, that's a good team to lose to.

Overall, it was a great experience. RadioShack did a great job on the kits, giving us everything we needed to complete the challenge, and FIRST did an equally great job on organizing the event. The boys on our team learned a lot about teamwork and engineering and had a blast in the process. Figure 7 shows our alliance and all three bots. Figure 8 shows Team #8 — once ranked 51 out of 52, but who made it all the way to the second place alliance! SV

BOLT-ON WHEEL HUBS

MANUFACTURER	Hub Name	Part Number	Image	Dimensions A	B	C	D
HUB STYLE MOUNTS							
Lynxmotion www.lynxmotion.com	Universal Hub — 6 mm	Hub-02	A	6 mm	0.37	0.87	0.11
	Universal Hub — 4 mm	Hub-03	A	4 mm	0.37	0.87	0.11
	Mounting Hub	Hub-05	B	6 mm	0.32 x 0.07	0.87	0.08
	Mounting Hub — 4 mm	Hub-06	C	4 mm	N/A	0.87	N/A
	Mounting Hub — 6 mm	Hub-07	C	3 mm	N/A	0.87	N/A
Team Associated www.teamassociated.com	LH Wheel Hub — Clamp	48655	D	0.25 in.	0.64	1.12	0.11
	Wheel Hub Left-Hand RC10L	8212	E	0.25 in.	0.64	1.12	0.15
Robot Store www.robotstore.com	Mounting Hub Black Nylon	386062	F	0.25 in.	0.64	1.18	0.08
HEX STYLE MOUNT FOR R/C CAR AND TRUCK TIRES							
DuraTrax www.duratrax.com	Drive Hub w/Wheel Nut	DTXC7510	G	6 mm	0.47	19 mm (0.75 in.) HEX	0.4

Note: All dimensions are in inches unless noted.

Image A

Image B

Image C

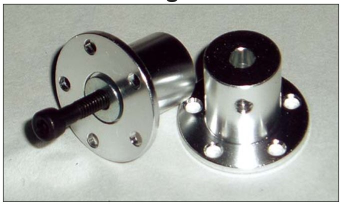


Image D

MATRIX

by Pete Miles

Upcoming topics include SBCs and H-bridges, sensors, kits, and actuators. If you're a manufacturer of one of these items, please send your product information to: **BrainMatrix@servomagazine.com** Disclaimer: Pete Miles and the publishers strive to present the most accurate data possible in this comparison chart. Neither is responsible for errors or omissions. In the spirit of this information reference, we encourage readers to check with manufacturers for the latest product specs and pricing before proceeding with a design. In addition, readers should not interpret the printing order as any form of preference; products may be listed randomly or alphabetically by either company or product name.

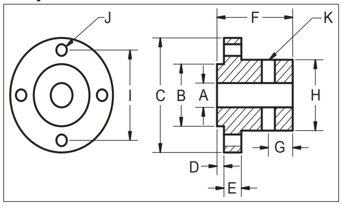
E	F	G	H	I	J	K	Material	List Price
HUB STYLE MOUNTS								
0.25	0.47	0.23	0.37	0.61	#5-40 Thread, four places	#10-32 Thread	Aluminum	$8.00/pair
0.25	0.47	0.23	0.37	0.61	#5-40 Thread, four places	#10-32 Thread	Aluminum	$8.00/pair
0.87	1.07	0.25	0.38	N/A	#8-32 Threaded Center Tap	#10-32 Thread	Aluminum	$10.00/pair
0.065	0.63	0.18	0.5	0.7	0.090 thru — five places and #5-40 Threaded Center Tap	#6-32 Thread	Aluminum	$8.00/pair
0.065	0.63	0.18	0.5	0.7	0.090 thru — five places and #5-40 Threaded Center Tap	#6-32 Thread	Aluminum	$8.00/pair
0.12	0.7	0.29	0.72	0.93	#4-40 Thread, four places	#4-40 Thread	Aluminum	$17.49/ea
0.12	0.9	0.14	0.55	0.93	#4-40 Thread, four places	#10-32 Thread	Aluminum	$5.79/ea
0.2	0.87	0.28	0.73	0.93	#5-40 Thread, four places	#10-32 Thread	Nylon	$7.99/pair
HEX STYLE MOUNT FOR R/C CAR AND TRUCK TIRES								
0.3	0.71	0.16	0.39	N/A	19 mm x 1 mm Thread	5 mm x .8 mm Thread	Aluminum	$4.99/ea

Image E

Image F

Image G

Defines what the dimensions are.



The Robosapien V2

by Jamie Samans

BIGGER AND BETTER

It's hard to believe it has been almost two years since the introduction of WowWee's groundbreaking Robosapien robot.

In that time, well over a million robots have found their way to the far corners of the globe. Mark Tilden — Robosapien's creator — even told me that there is a Robosapien at a research station on the continent of Antarctica!

The Robosapien was a huge success, a deceptively simple toy robot whose biomorphic roots were easy to see. What's more, with a little bit of creativity and a Phillips head screwdriver, the Robosapien was capable of making the transition from toy into tool, and has inspired countless hackers to use it as a platform for their projects. From university robotics labs to hobbyist workbenches in garages, the Robosapien (or as we will now refer to it, the RSV1) has been widely accepted as an excellent platform for modification.

Enter Robosapien V2, or RSV2, the next-generation bipedal robot from WowWee. Initially available only from the Sharper Image as an "Exclusive Signature Series" model (see Figure 1), I managed to get my hands on one of the first robots off of the production line and put it through the paces.

The first thing that you notice about RSV2 is how big it is. At 22", it is a full 10" taller than the RSV1. It is huge and very impressive looking and usually people's first response to it is a rather stunned " ... coooool." But does bigger really mean better? This is one of the main questions I hope to explore.

My initial impression is that RSV2 addresses many of the RSV1's shortcomings, as well as adding a few really neat features. However, these improvements and features come with some downsides, namely, the increased physical size, increased noise levels, and of course, an increased appetite for batteries. Also not to be overlooked is the price tag. With street prices expected to be in the $230-$250 range, the RSV2 is almost three times the cost of the RSV1.

Turning on the RSV2

Once you have extricated the RSV2 from the diabolical packaging system of tape, cardboard, and stiff wire that is used to hold it securely during shipping, the next task is to load it up with batteries. The RSV2 takes a grand total of 13 batteries, in the form of six "D" cells and seven "AAA" cells. Three "D" cells go into each foot and control the robot's body motors. Additionally, there is space for two "AAA" cells in each foot (see Figure 2), and these are used to power the brain.

Finally, the "video game controller" remote control takes three "AAA" cells. A nice touch: When the brain batteries become low, RSV2 will issue a warning and then shut down. A nicer touch would have been similar warnings for when the body and remote batteries become low! WowWee claims that the RSV2's batteries should last for about six hours with continuous use, and about 14 hours with light (i.e., tabletop) use.

Now that it is loaded with batteries, it's time to power on the RSV2. The on/off switch is located familiarly in the same spot as it was on the RSV1, on the left side of the robot's back. Upon powering it up, you are greeted with: "Self diagnostic initiated. I am WowWee Robotics model RSV2." That's right, this robot talks. Gone is the quaint "international caveman speech" of the RSV1. The voice is appropriately robot-like, but I swear I hear Mark Tilden's voice in there underneath all of the audio effects processing.

So, by this point you are finally beginning to adjust a bit to the size of the RSV2, and now that you have it powered up, what strikes you is the

noise. The RSV2 is just plain loud, and the voice is only part of this — its motors are also much louder than the smaller ones found in the RSV1. Since the RSV2 can move two motors simultaneously — for instance, you can raise both arms at the same time — the motor noise is sometimes doubled. Walking around on hardwood floors results in a lot of noise too, as the robot weighs around 11 pounds with batteries.

I should note that as I continue to use the RSV2, the noise doesn't seem to be bothering me as much. I don't know if it is because I am getting used to it, or if the motors needed a slight break-in period to "loosen up," or if it is some combination of the two. But it doesn't seem as noisy now as it did a week ago when I first got it. Another interesting point is that my wife thought I was crazy for saying that the RSV2 was really loud, she claims that "it sounds the way a robot is supposed to sound."

Operating the RSV2

The primary means for controlling the RSV2 is by using the video game style remote control (see Figure 3). Like RSV1, the RSV2 uses infrared signals, which means you will need line-of-sight, and if you have more than one RSV2, you will not be able to use them independently of each other, at least not while they are in the same vicinity. There are no IR command conflicts between RSV2 and RSV1 however, so these two bots can battle it out in the same room to your heart's content.

The controller takes some getting used to. At first I hated it, but with time I am learning how to use it to unleash the RSV2's full potential. I recommend a "cheat sheet" (see Figure 4) until you get comfortable with the controller setup. I feel that the controller is in no way as well laid out or as intuitive as RSV1's remote control, but what it lacks in elegance it more than makes up for in control.

Consisting of two "thumbstick" style joysticks and 12 buttons, the controller is capable of sending a total of 136 different commands to the RSV2. Walking is controlled by the left thumbstick; upper body and arm movements are controlled by the right thumbstick.

By using one or more of the three "shift" buttons on the controller's shoulders, you gain an incredible amount of flexibility. Using the shift buttons in concert with the buttons on the controller's face gives you access to the RSV2's "attitude" animations, as well as the programming modes.

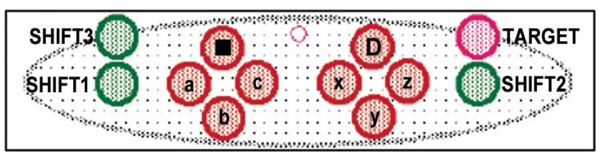
Using the shift buttons with the right thumbstick gives you fine-tuned access to controlling RSV2's upper body. On its own, the right thumbstick controls the head and upper body. Using the shift buttons allows you to control the robot's arms individually and together, to control just the RSV2's head, or just its hips, and so on. It seems clumsy at first, but as I got used to the controller, I discovered that it allows

Figure 1. RSV2's box is over 25 inches tall and has a shipping weight of 17 pounds.

Figure 2. One of the battery compartments. Note the two "AAA" cells at the top.

Figure 3. The RSV2 remote control.

RS2 M2 TX CHEAT SHEET:



RS2 (M2) Codes	R Stk	■	a	b	c	x	y	z	D
	HEAD & UPPER BODY	STOP	SOUND SENSE ON/OFF	VISION SENSE ON/OFF	POS PROG ENTER	POS PROG PLAY	GAIT CHNGE (RST)	FREE ROAM ON/OFF	DEMO 1 DANCE
	RIGHT ARM	RESET	‹RIGHT ARM THROW	‹RIGHT LOW PICKUP	‹RIGHT MED. PICKUP	‹RIGHT ARM GIMME	‹RIGHT ARM DROP	‹RIGHT BOWL	DEMO 2 STRTCH
	LEFT ARM	SLEEP/ WAKE	›LEFT ARM THROW	›LEFT LOW PICKUP	›LEFT MED. PICKUP	›LEFT ARM GIMME	›LEFT ARM DROP	›LEFT BOWL	LIE/ SIT/ RISE
	HIPS TWIST TILT	GET UP	‹RIGHT KICK	‹RIGHT PUSH	‹RIGHT HAND CHOP	›LEFT HAND CHOP	›LEFT PUSH	›LEFT KICK	FART
	BOTH ARMS	CLEAR PROG	SOUND PROG	VISION PROG	MAIN PROG	EXEC PROG	GUARD MODE	CLEAR ENTRY	HIGHS
	HEAD ONLY	BULL-DOZER FWD	LAUGH	INSULT	‹RIGHT PLACE	›LEFT PLACE	PLAN	SPARE CHNGE	HEY BABY
	HEAD & UPPER BODY	BULL-DOZER BWD	ROAR	DIODE	FETCH ME A BEER	DANGR	CALM DOWM	HUG	BURP
	UPPER BODY ONLY	POWER DOWN	PLAY WITH RS2	PLAY WITH RAPTR	PLAY WITH PET	MANUAL W-BAL: SUN	MANUAL W-BAL: DIM	MANUAL W-BAL: WHITE	OUCH!

Figure 4

Figure 4. This is Mark Tilden's own personal "cheat sheet" for the RSV2.

Figure 5

Figure 6

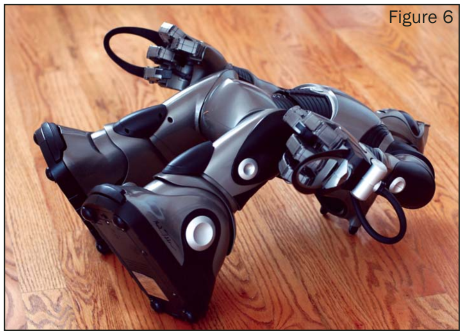
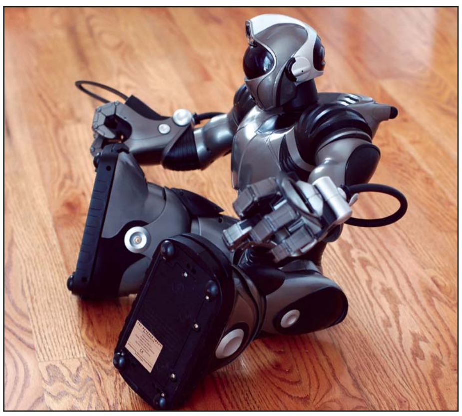


Figure 7

Figures 5-7. RSV2 can stand back up from a laying down position.

for quite a bit of precision. For example, the longer you hold the right thumbstick to raise, say, the right arm, the higher it will go. With PC control (or VERY nimble fingers), it is possible to have essentially centimeter precision. I felt that the lack of precise movement was one of RSV1's major shortcomings.

Figure 8. From a sitting position, RSV2's head and arms are still controllable.



Another area where RSV2 improves upon RSV1 is in its walking ability. RSV2 features four different gaits and — depending on your floor surface — does a much better job of walking in a straight line than the original Robosapien. I tested the RSV2 on a variety of floor surfaces in my house: hardwood floors, low shag carpet, "berber" carpet, and concrete floors. RSV2 does best by far on hard surfaces. It will walk just fine on carpeted surfaces, but it is significantly slower, and because RSV2 is a bit top heavy it tends to lose its balance more on carpet and will fall over. It's almost like the springy padding under the carpet throws off the resonance dynamics required for it to do things like spin in place and turn.

However, one of RSV2's coolest features is the ability to stand back up if it has fallen on its back (see Figures 5 through 7), although, of course, this works better on hard floors too. You can also make the RSV2 lie down on its back from a standing position, and from there command it to sit up (see Figure 8). From a sitting position, you still have full control of its head and arms.

Figure 9. RSV2 performs a karate chop on an unsuspecting RSV1. Look at the size difference!

What would a Robosapien be without attitude? The RSV2 has the same bad manners, rowdy personality, and kung fu abilities as RSV1, although it is able to articulate things better due to the fact that it is no longer burdened by the language barrier inherent in "international caveman speech." This can be either a plus or a minus, depending on your outlook. All of the old favorites are here, including an updated dancing demo, and quite a few new ones, as well.

One that is particularly notable is "Get your own drink" — entitled "fetch" in the manual. According to Mark Tilden, the RSV2's hands were originally designed to carry and lift a full can or bottle of your favorite beverage, but this feature had to be removed due to safety reasons. In the retail version, RSV2 is able to lift and carry things up to about five ounces. The hands do fine carrying empty bottles and cans, so the design was left in place.

At any rate, RSV2 has received a significant upgrade in its abilities to manipulate objects in its environment. It can take objects from you, give them back, pick them up from two different heights, throw them, even push and pull. I am not very good at it yet, but with some practice I think that RSV2 will be capable of a fair amount of precision. Speaking of precision, Mark Tilden tells me that RSV2's karate chops are "designed to bonk RSV1 right on the noggin" (see Figure 9).

In addition to the boisterous personality skits and increased manipulation capabilities, the RSV2 also has the ability to interact with the Roboraptor, the Robopet, and even another RSV2. I wasn't able to test these features, but it appears that the RSV2 has some limited ability to send out IR commands to interact with these other robots. This is extremely interesting from a hacker's perspective. Using the RSV2 to blast out IR signals to control an RSV1 or even a Roomba robotic vacuum would be insanely cool.

Vision and Sensors

Probably the biggest addition to the RSV2 is its vision system (see Figure 10). Consisting of an infrared radar system and a color camera, RSV2 is able to avoid obstacles, track movement, and recognize some limited colors.

RSV2's obstacle avoidance abilities are fantastic, and this is best seen in the robot's autonomous "free roam" mode. Unlike RSV1, where you had to program a complicated sequence of moves to put the robot into a semi-autonomous mode, and then wait for its finger or toe sensors to physically hit something to trigger an avoidance, RSV2 enters "free roam" mode with the touch of a single button. It does a really good job wandering around on its own, using a combination of the IR system and its touch sensors. Note that due to the size of the RSV2, you need a fairly large area for this to work. For instance, my office — which has an open area of about 5' x 6' — is too small. It will also not recognize drops, such as stairs or the edge of a table it is standing on.

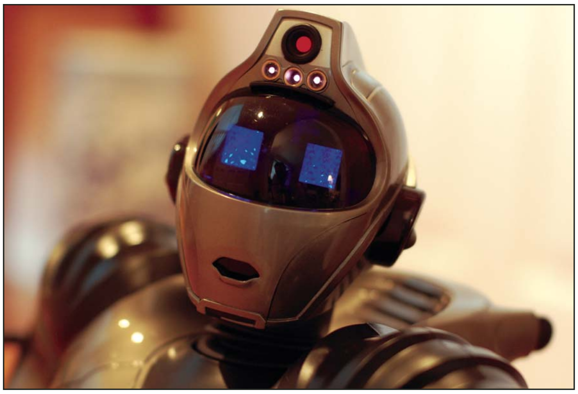


Figure 10. A close-up showing RSV2's vision system.

The IR system allows RSV2 to "track" objects in its field of vision. Its head and upper body will follow whatever it has drawn a bead on. If the object that it is tracking is close enough and becomes stationary, RSV2 will ask you to hand it to him. This is downright uncanny. This system also works in conjunction with RSV2's color camera. RSV2 can recognize green, red, blue, and flesh tones.

The robot comes with a green ball and a set of three red bowling pins, and can actually take the ball, tell you about its "itchy bowling arm," find the pins (if you set them up in front of the robot), and then throw the ball at them. Its aim isn't great, but when it connects and bowls a strike, it is fantastic. If there is a downside to the color vision system, it is that it requires a lot of light to operate. RSV2 comes with adjustable lighting settings (daylight, indoor yellow, and indoor white), but these really don't help very much. I think some of the first RSV2 hacks we will see will be augmented lighting systems.

RSV2's sensor system is significantly upgraded over the one found in RSV1. RSV2 has a visual sensor, a stereo sonic sensor that can tell which side the sound is coming from, heel and toe sensors on its feet, grip sensors inside its hands that can detect if it is holding an object or not, and "gauntlet" sensors at the wrist (see Figure 11). It also has a tilt sensor that detects when the robot has fallen over.

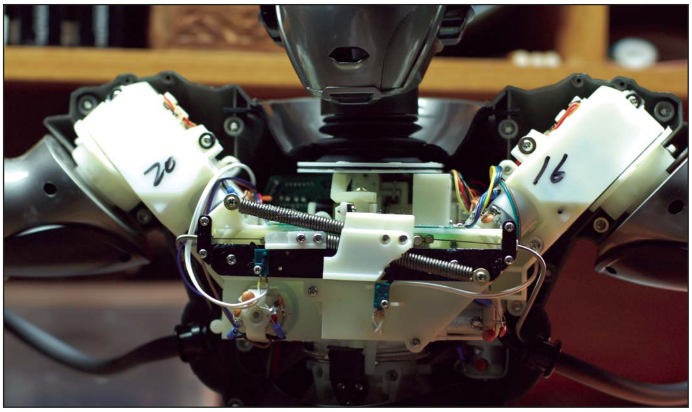
Programming

In programming the RSV2, again we see a system that is in many ways very similar to the RSV1, but with significant upgrades and additions. The most noticeable upgrade to RSV2 is the ability to save your programs in memory, even after you turn the RSV2 off.

Besides the memory upgrade, RSV2 also has an entirely new programming system (in addition to reactive sonic and vision programs, and a main program mode). This new programming system is called "puppet mode" and allows you to program the RSV2 manually by moving its upper body into different positions. You can also combine upper and lower body movements (such as walking, turning, or even karate kicking) by using the foot sensors to input moves.

Figure 11. RSV2's "gauntlet" touch sensor on the robot's hand.

Figure 12. Inside RSV2's chest cavity. Note the silver "tendon" — this is linked to the side-to-side movement of the head and controls the opening and closing of the hand.



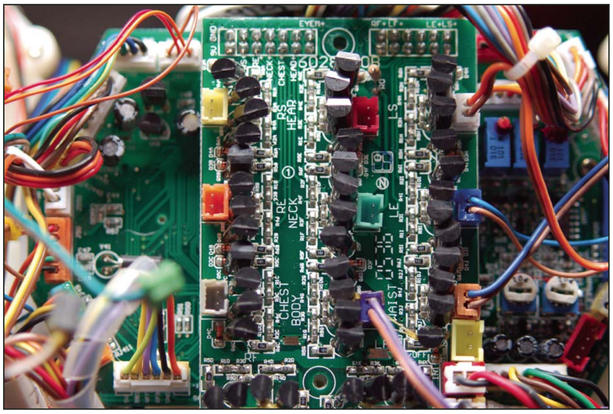


Figure 13. This photo shows RSV2's driver board. It would appear that the 44 transistors make up 11 H-bridges to control the 11 main motors. RSV1 had a dedicated motor driver chip.

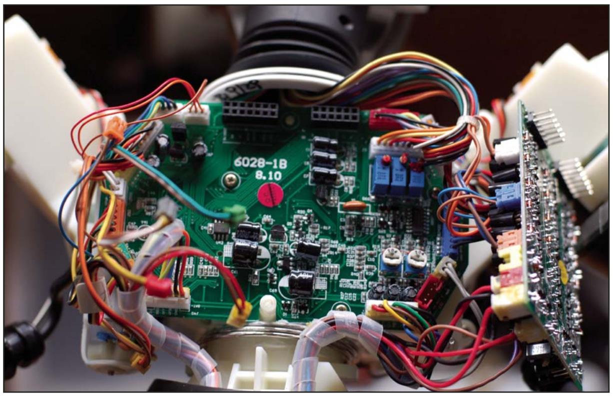


Figure 14. This is the main circuit board which resides underneath the driver board. The underside (not shown) contains two chips; both are covered in an epoxy blob like the CPU for RSV1.

Like a lot of the new things with the RSV2, puppet mode takes some practice to perfect. Puppet mode has three individual modes: main positional, which holds 14 steps (though the manual says 12), and left and right positional which hold five steps each and are activated by quickly pressing one of the foot or gauntlet sensors twice.

Beyond this, programming the RSV2 is very similar to the RSV1, but with more available steps. The main program mode and both reactive programs (sonic and vision) each hold 20 steps. This is significant, as the RSV2 has the ability to use conditional subroutines just like the RSV1. Meaning, you can program a 20-step routine into the sonic sensor, and then fill all 20 steps of the master program with your sonic program routine, resulting in a 400 step program! But it gets even better.

You can fill the reactive program modes with subroutines that you have stored in the main puppet mode program (press the "x" button on the remote to store a positional program as a single call). The result of programming all of these subroutines simultaneously is 14 x 20 x 20 or a 5,600 step program. This is not mentioned in the manual, for the obvious reason that in executing a 5,600 step program, the batteries (or the motors and/or motor drivers) would probably die before it was able to complete it. The author strongly recommends that you do not try this with your RSV2 ... but it is nice to know. Think of it like the speedometer in your car: mine goes up to 200 mph, but I would never drive the car that fast!

Hackability

Hackability of the RSV1 was one of its biggest draws. Over the past year and a half we have seen basically four types of hacks to the RSV1: cosmetic, enhancement, brain replacement/augmentation, and bolt-on peripherals. All four will be possible with the RSV2, and with its vision system, color camera, improved sensors, and more precise movement capabilities, I think we will see quite a few interesting projects based on the RSV2.

For example, Mark Tilden told me that 56 MHz DSP doing all of the color-blob image processing has a USB port. Another interesting hack would be to add a motor to each hand to optimize pick up speed and reliability (it currently uses a tendon system based on head movement — see Figure 12). Additionally, since RSV2 uses IR, it should be easy enough to port over the new commands and use a PC or a PDA to control the robot.

These hacks will take time, but remember that it took almost six months before we started seeing some of the better hacks for the RSV1. Another obstacle to hacking the RSV2 could be the complexity (see Figures 13 and 14). Although like RSV1, RSV2's boards are labeled reasonably well, and many of the connections are socketed, so this may or may not be an issue. Time will tell. The main obstacle I see to hacking the RSV2 is the cost. It's one thing to fry a $69 RSV1; it is another thing altogether to accidentally fry a $250 robot.

Conclusion

In many ways, RSV2 is very similar to its predecessor. From the diabolical packaging system used to keep the robot in place during shipping, to its attitude and bad manners, from the conditional programming logic to the nicely labeled circuit boards, there is no mistaking that RSV1 and RSV2 are related. Does RSV2 have some drawbacks? Of course. I think that the cost, size, battery consumption, and lack of any sort of connectivity hurts it. To some, the similarity to RSV1 — in the sense that they are both programmable remote control toys — might also seem like a negative.

So is bigger really better? I think that it is. I feel that the RSV2 is a worthy successor to the RSV1. WowWee seems to have listened to a lot of the complaints with the RSV1 and remedied them with this release, and they threw in some cool new features, too. Like RSV1, it will take some creativity and curiosity to transform RSV2 from toy into tool. But when has that ever been a bad thing? SV

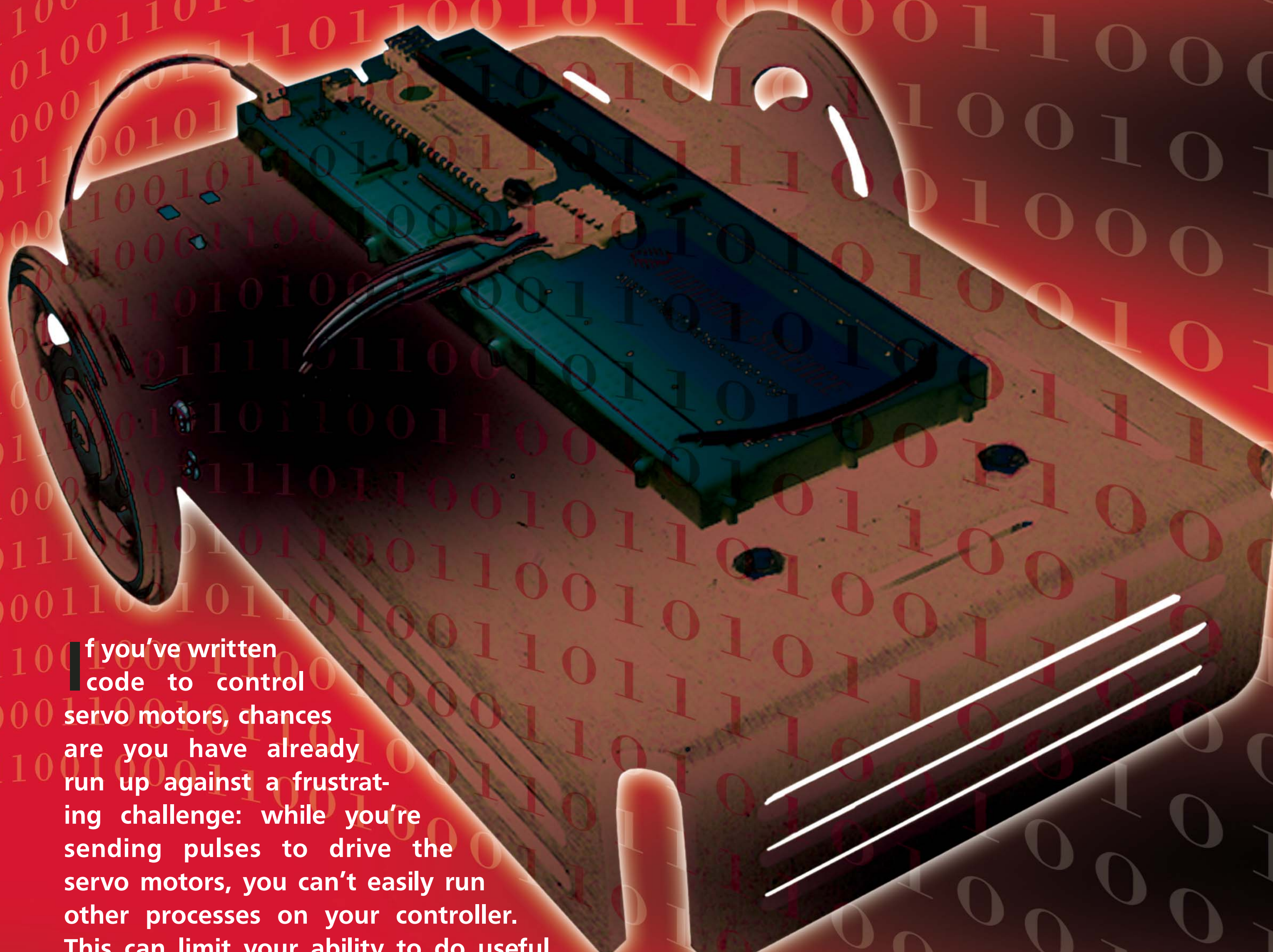
About the Author

Since winning SERVO Magazine's "Hack-A-Sapien" contest in 2004, Jamie Samans went on to write the Robosapien Companion: Tips, Tricks, and Hacks, published last summer by Apress. Featuring a lengthy interview with Robosapien's creator, Mark Tilden, as well as exclusive development photographs from Tilden's own personal collection, the book looks at not only where you can go with your Robosapien, but where Robosapien comes from.

by Sam Christy

Using Interrupts to Control Servos

Stop What You're Doing and Start Interrupting!



If you've written code to control servo motors, chances are you have already run up against a frustrating challenge: while you're sending pulses to drive the servo motors, you can't easily run other processes on your controller. This can limit your ability to do useful things, such as checking sensor values, performing computations, and controlling other output devices.

Yes, it's possible to do these things between the pulses that you send to the servo motors, but the results are often unsatisfying. If you attempt any complicated multitasking, your servos may even stutter, instead of turning smoothly, and your robot may run more like an old jalopy than a high-performance sports car.

The Basics

An **interrupt** is a change in the flow of code execution that is triggered by a microcontroller's internal hardware or by an external event, such as a change in a port value. **Timer overflows** — which occur when an internal timer reaches its maximum value — are commonly used as interrupt triggers. Since these timer-based interrupts occur at regular intervals, they are ideal for servo motor control, which requires very precise timing.

When an interrupt is triggered, code execution is suspended, an interrupt service routine is performed, and then code execution resumes at the "interrupted" point. Although interrupt service routines can take many forms, you can think of them as functions that are called automatically by the microcontroller. For the purpose of driving servo motors, the interrupt service routine is used to set certain microcontroller pins high or low, generating the pulse series needed for servo motor control.

Typically, continuous rotation servos are driven by a series of five-volt pulses, each lasting 1,000 to 2,000 microseconds, with 20 milliseconds between the start of each pulse. **Figure 1** illustrates this typical pulse series. The pin connected to the servo goes high, stays high for 1,000 to 2,000 microseconds, goes low, and then stays low until the start of the next pulse.

Delay routines are often used to control the duration and frequency of the servo pulses, resulting in wasted processing time, as shown in **Figure 2**. If you use delay routines to control the pulse timing, you have only intermittent opportunities to run other processes, and if these other processes take more than a few milliseconds to complete, they can adversely affect the servo pulse series, producing the aforementioned servo performance problems. With interrupts, you can run other processes continually, and even complex functions won't disrupt the pulse series.

The solution to this problem is simple: interrupts. With interrupts in your code, you can send a continuous series of precisely timed pulses to the servo motors, while running other processes at the same time. Interrupts have a reputation for being difficult to understand and debug, and this reputation is partly deserved. But with patience and a little practice, even novice programmers can get up to speed and start using interrupts to control their servos!

Understanding Timers

Since you will be using timer-based interrupts to control your servos, it is essential that you understand how the microcontroller's internal timers work. The timers increase in value at a rate that is tied to the microcontroller's raw processing speed. For example, if your microcontroller executes one instruction per microsecond, each of the chip's timers will increase in value by 1 every microsecond until it reaches its maximum value and resets to 0. In practice, timers are often adjusted so that they increase in value at a slower rate — a procedure called pre-scaling. Pre-scaling by a factor of 2 slows the timer by a factor of 2, pre-scaling by a factor of 4 slows the timer by a factor of 4, and so on.

A timer's interrupt service routine is called whenever the timer overflows, i.e., whenever it reaches its maximum value and resets to 0. For example, the interrupt for an eight-bit timer is called whenever the timer's value cycles from 255 to 0. By pre-scaling the timer for your interrupt, you can control how frequently the timer's interrupt service routine is called. At the instant the timer overflows, another event occurs: a bit associated with the timer — the timer's interrupt flag — is set to 1. The flag can be used to confirm that the timer has overflowed.

The Code

This section provides step-by-step instructions for writing C code to control a robot with two servo motors, using interrupts. Your code will vary slightly depending on the compiler and chip you are using. All of the code shown in this article was developed using the on-line development environment created by my company, Machine Science, Inc., for Microchip's PIC16F877 microcontroller. The underlying compiler is the free demo version of the Hi-Tech PICC Compiler. Figure 3 shows the completed robot.

Because the code can be a little confusing, we've broken it down into five steps: **1)** configuring the chip and declaring variables; **2)** enabling the interrupts and setting up output pins;

FIGURE 1. A typical servo motor pulse series.

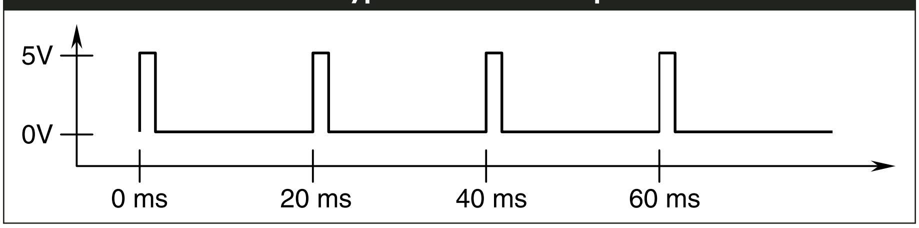
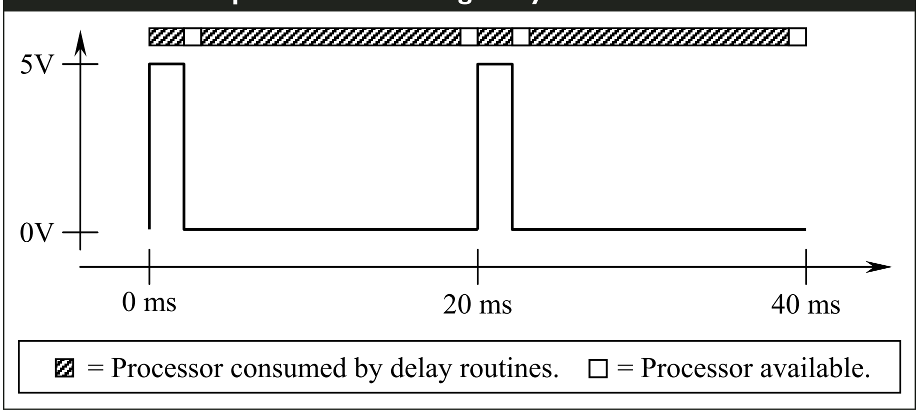


FIGURE 2. The problem with using delay routines for servo control.



3) pre-scaling the timer; **4)** writing the interrupt service routine; and **5)** putting it all together.

STEP #1: Configuring the Chip and Declaring Variables

Listing 1 shows the code required to configure the PIC16F877 and declare the necessary variables. The CONFIG statement establishes important hardware settings for the operation of the microcontroller. The left_motor variable will control the duration of the pulse sent to the left motor, right_motor will control the duration of the pulse sent to the right motor, and counter will be used to control the frequency of the pulses. Because these variables need to be available in your main function as well as your interrupt service routine, they must be declared as global variables near the top of your code file.

STEP #2: Enabling the Interrupts and Setting Up Output Pins

To enable interrupts on the PIC16F877, you need two separate statements in your main function: one to enable interrupts globally and one to enable interrupts for the specific timer you're using. As noted above, different chips may require slightly different interrupt enable statements. Check the interrupt logic table in your chip's datasheet for specific guidance. In addition, you will need to set up two microcontroller pins as outputs. Listing 2 shows the required code for the PIC, assuming that the servos are connected to Ports B0 and B1.

STEP #3: Pre-Scaling Your Timer

In setting up the hardware for our project, we connected the PIC16F877 to a 4 MHz oscillator, which gives the chip a raw processing speed of one million instructions per second — or one instruction every microsecond. We then pre-scaled the timer for our project, Timer0, by a factor of 2 (which is the minimum possible pre-scaling factor for the PIC). Configured this way, Timer0 increases in value 500,000 times per second — or once every two microseconds. Since Timer0 is an eight-bit timer, with a maximum value of 255, it overflows every 512 microseconds. The pre-scaling of Timer0 is accomplished by setting certain registers in your main function, as shown in Table 1.

Listing 3 shows the code needed to pre-scale Timer0 by a factor of 2.

TABLE 1. Pre-Scale Values for Timer0 on the PIC16F877.

Pre-Scale Factor	Value	PS2	PS1	PS0
2	000	0	0	0
4	001	0	0	1
8	010	0	1	0
16	011	0	1	1
32	100	1	0	0
64	101	1	0	1
128	110	1	1	0
256	111	1	1	1

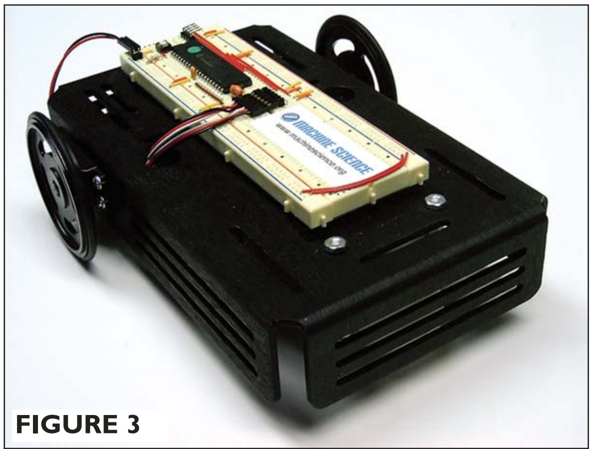


FIGURE 3

A Machine Science robot with two servos.

STEP #4: Writing Your Interrupt Service Routine

With your interrupts enabled and your timer pre-scaled, you're ready to write your interrupt service routine. The purpose of the routine is to generate the pulse series needed to drive your servo motors. To do this, you can make use of the fact that the Timer0 interrupt is triggered every 512 microseconds. Therefore, the routine should set the pins connected to the servos high once every 39 times that the interrupt is triggered (39 * 512 microseconds = 19,968 microseconds), make them stay high for two to four interrupt calls (2 * 512 microseconds = 1,024 microseconds and 4 * 512 microseconds = 2,048 microseconds), and then make them go low. In the example shown in Listing 4, the left servo motor is connected to Port B0 on the microcontroller, and the right servo motor is connected to Port B1.

LISTING 1. Configuring the chip and declaring variables.

```
#include <pic.h>

__CONFIG(0X7E61);            //Recommended configuration for PIC
char left_motor;             //Declare a global char variable left_motor
char right_motor;            //Declare a global char variable right_motor
char counter;                //Declare a global char variable counter
```

LISTING 2. Enabling interrupts and setting up output pins.

```
char counter;                //Declare a global char variable counter

void main(void)
{
   GIE=1;                    //Enable global interrupts
   T0IE=1;                   //Enable Timer0 interrupts
   TRISB0=0;                 //Set up Port B0 as an output
   TRISB1=0;                 //Set up Port B1 as an output
}
```

LISTING 3. Pre-scaling the internal timer.

```
   TRISB1=0;                 //Set up Port B1 as an output
   T0CS=0;                   //Set Timer0 to instruction clock
   PSA=0;                    //Assign pre-scaler to Timer0
   PS2=0;                    //Set first digit of pre-scale value
   PS1=0;                    //Set second digit of pre-scale value
   PS0=0;                    //Set third digit of pre-scale value
}
```

LISTING 4. Writing the interrupt service routine.

```
char counter;                     //Declare a global char variable counter

void interrupt servo(void)        //Define interrupt routine called servo
{
    if(T0IF==1)                   //Check if Timer0 has overflowed
    {
        T0IF = 0;                 //Reset Timer0 flag
        counter++;                //Increase counter by 1
        if(counter==39)           //Check if interrupt was triggered 39 times
        {
            RB0=1;                //Start pulse to left servo motor
            RB1=1;                //Start pulse to right servo motor
            counter=0;            //Reset counter
        }
        if(counter==left_motor)   //Check if time to end left servo pulse
            RB0=0;                //End pulse to left servo motor
        if(counter==right_motor)  //Check if time to end right servo pulse
            RB1=0;                //End pulse to right servo motor
    }
}
```

LISTING 5. Finishing touches.

```
void main(void)
{
        PS0=0;                    //Third digit of pre-scale value
        while(1)                  //Set up infinite loop
        {
            left_motor=2;         //Send 1024-ms pulse to left motor
            right_motor=4;        //Send 2048-ms pulse to right motor
        }
}
```

Resources

Microchip Technology
www.microchip.com

Hi-Tech Software
www.htsoft.com

Machine Science, Inc.
www.machinescience.org

STEP #5: Putting It All Together

Now, the rest is easy! You simply need to set the value of left_motor and right_motor in your main function, and these global variables will be available to your interrupt service routine whenever it is called. In a sense, you are "passing" these values to the interrupt function. Remember that a value of 2 will yield a pulse duration of roughly 1,000 microseconds, a value of 3 will yield a pulse duration of roughly 1,500 microseconds, and a value of 4 will yield a pulse duration of roughly 2,000 microseconds. (*Note that, since these values are not exact, you may need to manually adjust the center point on your servos in order to get them to stop when left_motor and right_motor have a value of 3. Most servo motors have an adjustment screw for this purpose.*)

Enjoying Your New Found Freedom

Consider what you have accomplished. Your interrupt function takes care of driving your robot's servo motors, freeing you to do more interesting things in your main function. Now you can add code for sensors, lights, speakers ... whatever you can imagine, without worrying about servo control. All the while, your interrupt function will be continuously sending pulses to the servos, keeping your robot on the move!

For more guidance on using interrupts to control servo motors, see Myke Predko's excellent book, *Programming Robot Controllers*, in McGraw Hill's Robot DNA series (ISBN 0-07-140851-7). **SV**

KIT KOLLEGE WITH PROFESSOR PROCHNOW

A Course on How to Build, Program, and Hack a Robot Kit

Up until this lecture, Kit Kollege has dealt exclusively with rolling robots, except for the JoinMax Smart Arm in Kit Kollege #2 (*SERVO Magazine* 09.2005). In this lecture, we are going to build a simple four-legged walking robot.

The JoinMax Digital Quadruped Robot Kit (JM-4FT-001; **www.mciirobot.com** and **www.robotplayer.com**) uses eight servo motors for achieving two degrees of freedom on each of its four legs. Orchestrating these movements is the same servo controller circuit board (JM-SSC16) that was found on the previously mentioned JoinMax Smart Arm kit. Based on an Atmel ATmega16 AVR eight-bit RISC microcontroller, this robot brain is able to deftly cycle through the complex raise, move, and lower sequences on all four legs enabling Quadruped to briskly walk about the floor.

Programming for the servo controller circuit board is provided with the included Windows-only program, *Mini Servo Explorer*. As discussed in Kit Kollege #2, this software generates a movement file that is played by the robot. In other words, there are no logic statements in the Quadruped program, only a series of servo power, speed, and position parameters.

While the Quadruped is tethered to your PC, you can adjust and fine-tune these parameters for the optimal walking gait. Then, after the servo controller circuit board is disconnected from your PC, these parameters are "played" by Quadruped.

The two most obvious hacks for Quadruped are the incorporation of a sensor into the robot and the replacement of the servo controller circuit board with a higher level programmable microcontroller. This type of obstacle avoidance and interpretive decision branching is essential for implementing autonomous movement into this robot.

For owners of the JoinMax Smart Arm kit, however, there is the opportunity to hack the Arm's gripper mechanism to the front of Quadruped. This hack is extremely easy and satisfying due to the simplified "plug-n-play" jumper ports on the servo controller circuit board.

In other words, just plug the Smart Arm gripper mechanism's servo connection cable into an available jumper port, set the servo parameters with *Mini Servo Explorer*, and Quadruped will be able to walk about and grab small objects. Furthermore, this hack can be readily adapted to the previously mentioned programmable microcontroller/sensor hack making Quadruped an autonomous walking robot with the ability to interact with its environment.

Class dismissed. **SV**

THIS MONTH:

LECTURE 7:

JoinMax Digital Quadruped Robot Kit

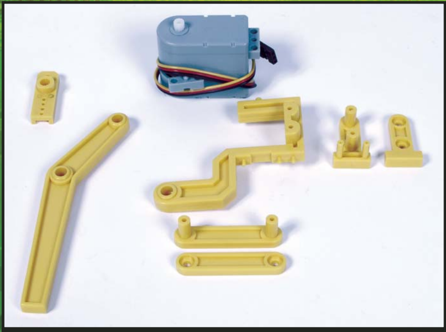
HACKS AND MODS

JoinMax Quadruped

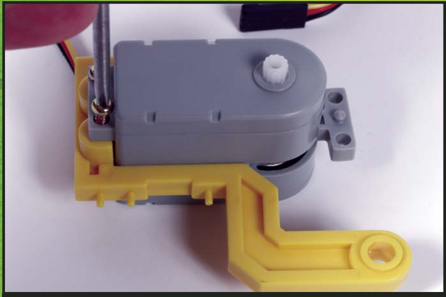
- Installation of the gripper mechanism from JoinMax Smart Arm on Quadruped "head."
- Gently shape the battery power cable wires for joining the servo controller circuit board to the battery box.
- Substitute alkaline batteries for rechargeable batteries.
- Replace the servo controller circuit board with a higher level programmable microcontroller (e.g., BASIC Stamp, BASICX, or Brainstem).
- Add touch or IR sensor.

During the assembly of Quadruped Robot Kit, changes/mods/hacks were employed for streamlining the assembly process, as well as enhancing the final robot's performance.

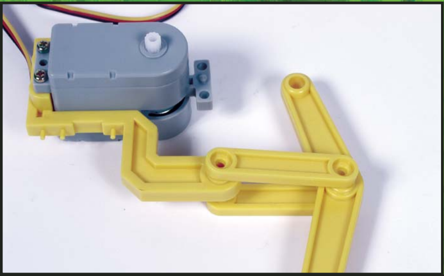
STEP 1. The JoinMax Digital Quadruped Robot Kit consists of eight servo motors, over 170 parts, and the *Mini Servo Explorer* software CD-ROM disc. All that you'll need for assembly is a No. 0 Phillips head screwdriver.



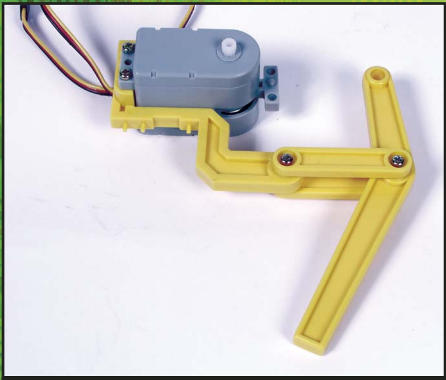
STEP 2. Begin construction by making four vertical leg-lifting servo assemblies. You'll build two right legs followed by two left legs.



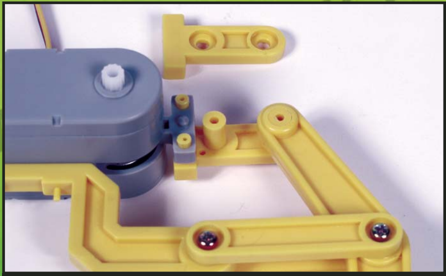
STEP 3. Attach the turning board to the first servo.



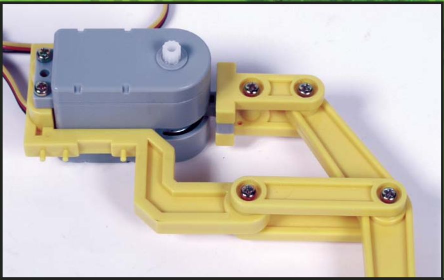
STEP 4. Use parts lower limb A and B to fix the leg to the turning board/servo assembly.



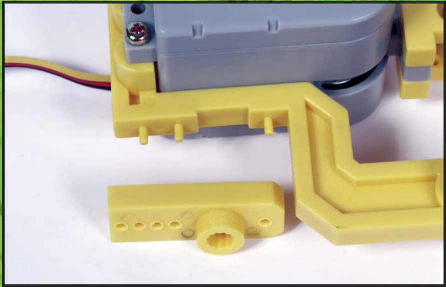
STEP 5. Make sure that the recessed socket of lower limb A receives each screw head.



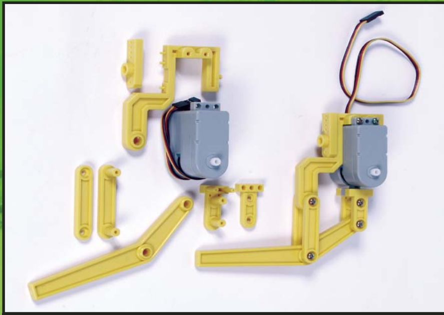
STEP 6. Connect the upper leg to the servo by attaching parts upper limb A and B to the servo's armature.



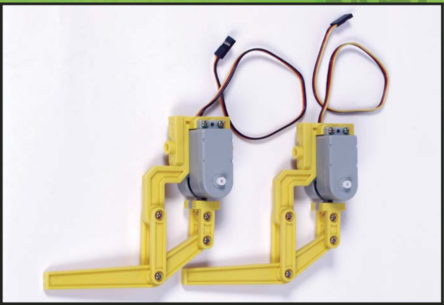
STEP 7. Tighten the upper limb screws securing the upper leg and servo armature.



STEP 8. Snap the servo motor turning arm socket under the turning board.

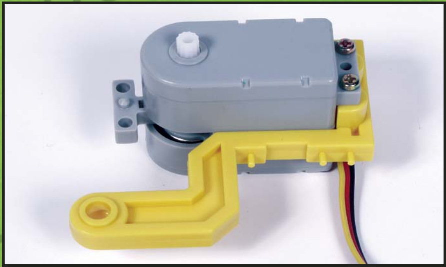


STEP 9. Build the second right leg exactly like the first one.

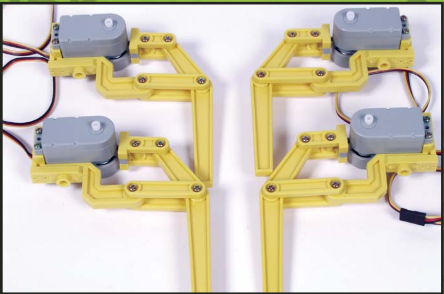


STEP 10. Lay the two completed right legs aside. Make sure that you label these legs for attachment to the right side of the Quadruped body.

STEP 11. Build two left legs following the same basic procedure that was used for assembling the two right legs.

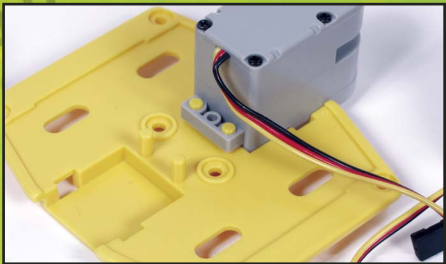


STEP 12. In order to ensure that you build two left legs, make sure that you observe the "flipped" orientation of the turning board. By carefully noting this fundamental difference, you will be guaranteed of building two left leg mates for the previously assembled right legs.

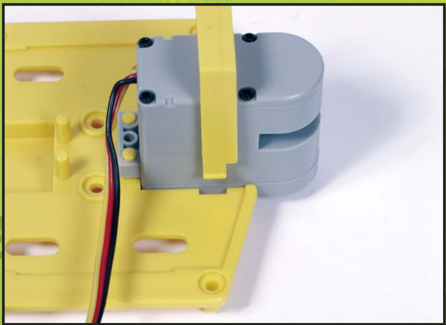


STEP 13. All four legs are complete. Make sure that you label and segregate the two rights legs from the two left legs and lay these assemblies aside.

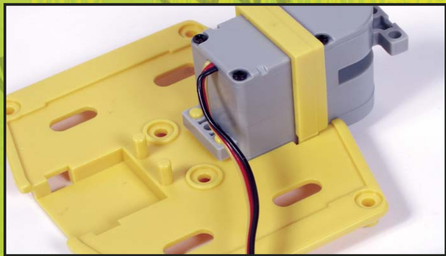
STEP 14. Build two bottom board assemblies. Each bottom board will hold two horizontal movement servos: one right leg and one left leg.



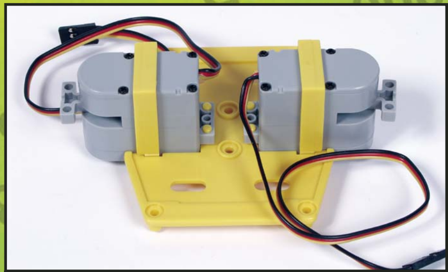
STEP 15. Align the first horizontal movement servo with the pins on the bottom board and snap the servo into place.



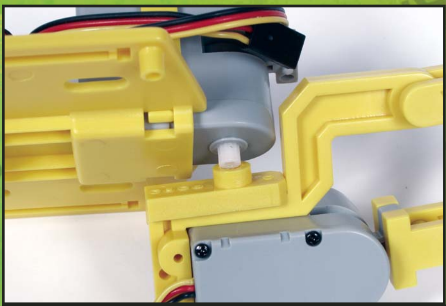
STEP 16. Fix the horizontal movement servo in place with the servo hoop hold-down. This hold-down snaps into two slots on either side of the servo.



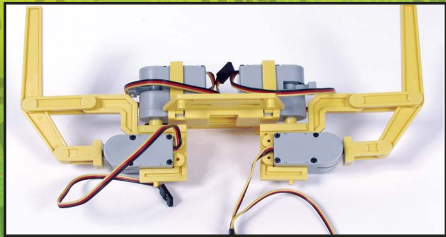
STEP 17. Carefully route the horizontal movement servo connection cable away from the bottom board.



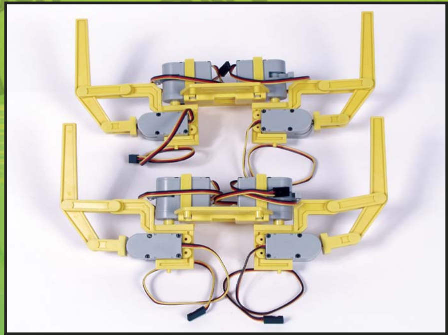
STEP 18. Attach the second horizontal movement servo to the bottom board.



STEP 19. Gently push each vertical lift leg assembly onto the horizontal movement servo drive gear. Specifically, you want to mate the servo motor turning arm socket on the vertical lift leg assembly onto the horizontal movement servo drive gear.

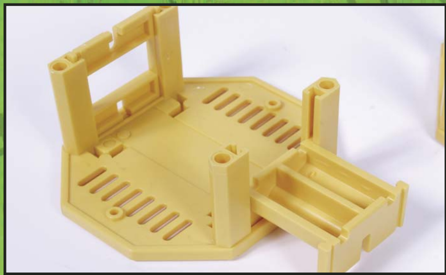


STEP 20. One bottom board assembly is complete. Set this module aside.



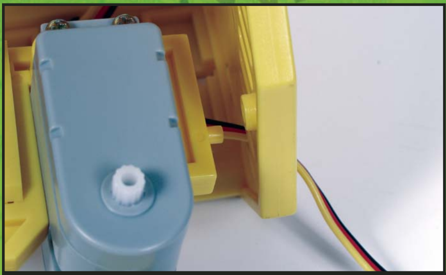
STEP 21. Both bottom board assemblies are now complete. Set both of these modules aside for now.

STEP 22. Snap the main body together.



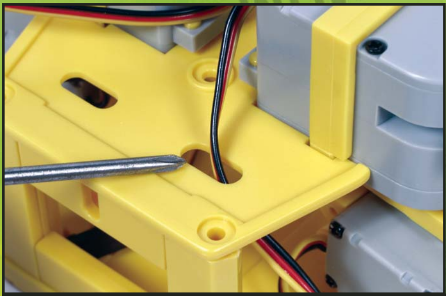
STEP 23. Carefully observe the orientation of the connecting board and the center waist part. This waist part will act as a conduit for holding the servo connection cables during the final assembly steps.

STEP 24. Lay the completed main body assembly on a flat surface.

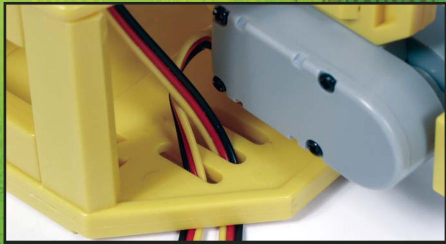


STEP 25. Carefully drop each bottom board assembly down onto the main body assembly. Make sure that the turning board pins are aligned with the corresponding receptacles on the main body assembly.

STEP 26. Route the servo connection cable through the slots in the main body assembly. The vertical lift servo cables from one half of the main body assembly should be inserted into the first forward slot on the body's front upper surface. This portion of the main body assembly will be the "head" for the Quadruped.



STEP 27. Each horizontal movement servo connection cable is routed through the adjacent slot in the bottom board.



STEP 28. The servo connection cable from each horizontal movement servo is then inserted through the second slot on the main body assembly. This second slot is behind the slot used by the vertical lift servo cable.

ROBOT KIT SOURCES

You can purchase Quadruped from any of the following sources. Please refer to each website for updated pricing information.

British Robotics
www.britishrobotics.com

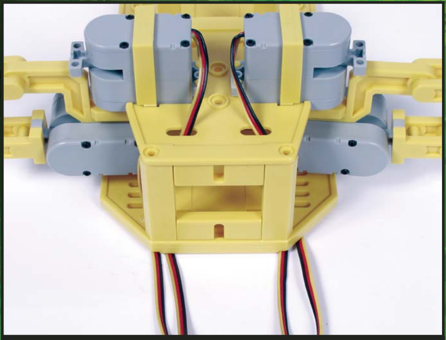
Cyber Hardware & Software
www.cyberhs.it

E-Clec-Tech
www.e-clec-tech.com

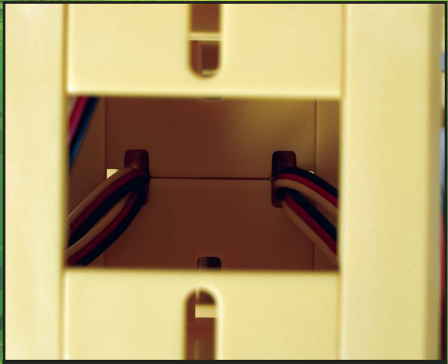
Garage Technologies, Inc.
www.garage-technologies.com

MCII Robot
www.mciirobot.com

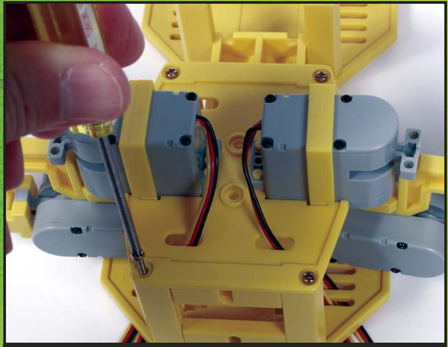
Pololu Corporation
www.pololu.com



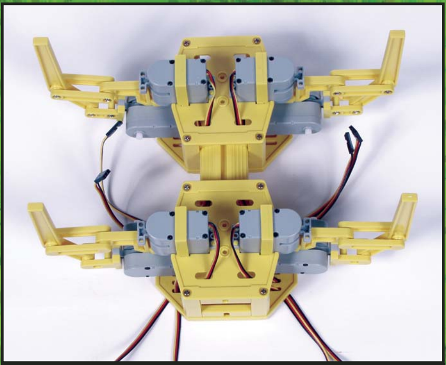
STEP 29. The “head” (i.e., the portion of the main body assembly that will hold the servo controller circuit board) of Quadruped is now wired.



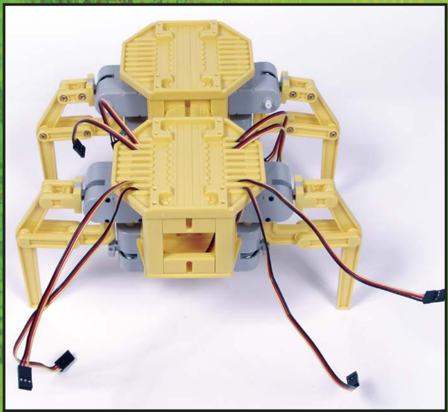
STEP 30. Carefully route the servo connection cables from the rear battery holding portion of the main body assembly through the waist conduit.



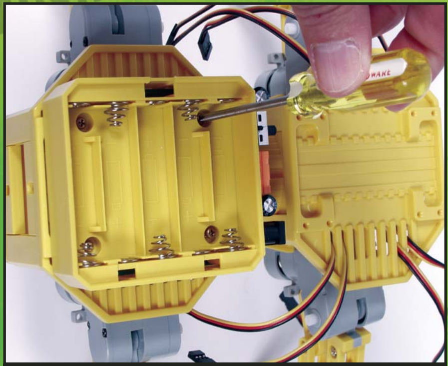
STEP 31. Complete the main body assembly by fixing the bottom boards with four screws each. Make sure that none of the servo connection cables are pinched inside the waist conduit.



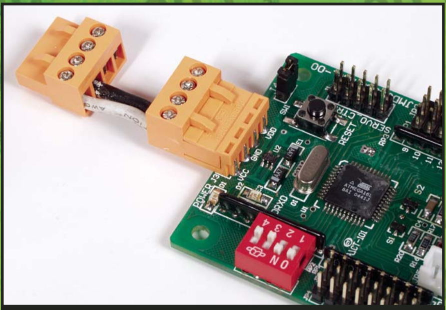
STEP 32. The completed physical assembly of the Quadruped main body.



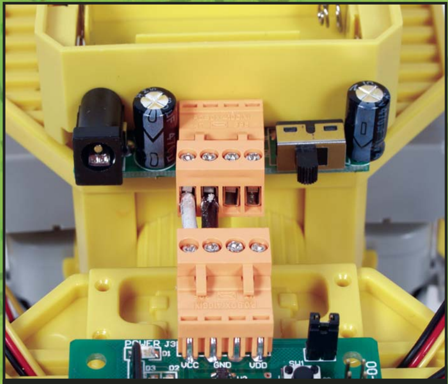
STEP 33. Flip Quadruped over. The main body portion with the servo connection cables is the “head,” while the rear portion will house the battery box.



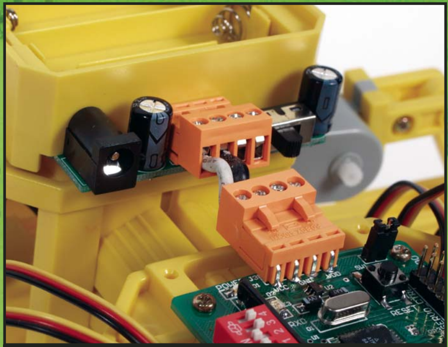
STEP 34. Attach the battery box to the rear of the main body assembly.



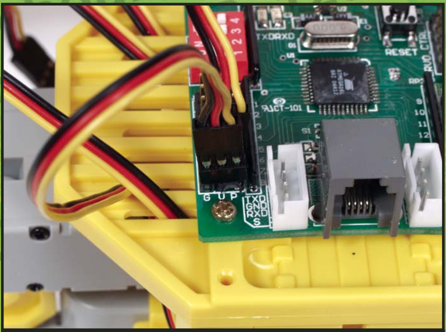
STEP 35. Prior to attaching the servo controller circuit board to Quadruped, plug the battery power cable into the circuit board’s power receptacle.



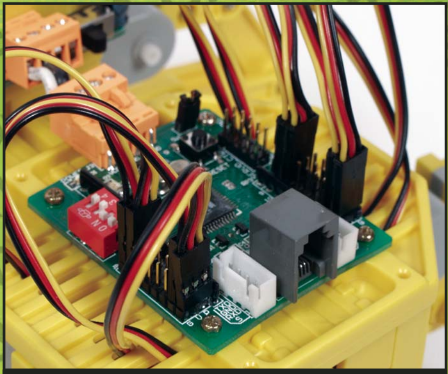
STEP 36. Snap the battery power cable into the battery box’s power receptacle.



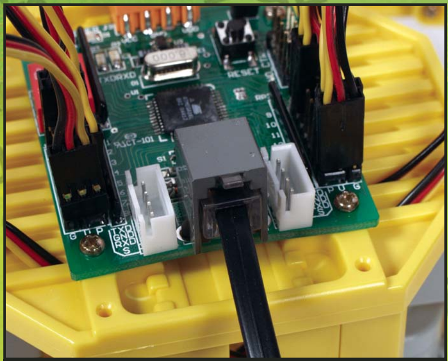
STEP 37. Gently bend the two power supply wires of the battery power cable upward for aligning the servo controller circuit board with the Quadruped head and use four screws to fix the circuit board to the head.



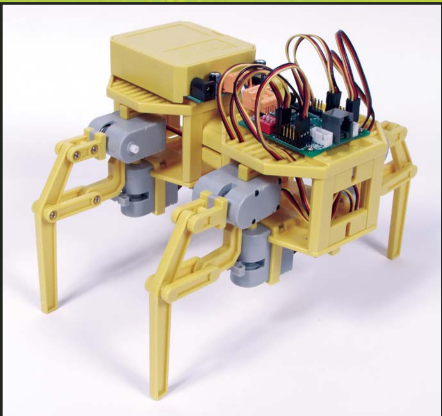
STEP 38. The eight servos are connected to the servo controller circuit board as: right rear leg (jumper port 1 and 2), right front leg (jumper port 7 and 8), left rear leg (jumper port 9 and 10), and left front leg (jumper port 15 and 16). In each case, the vertical lift servo connection is inserted in the first jumper port followed by the respective horizontal movement servo (e.g., 1 = right rear vertical lift servo and 2 = right rear horizontal movement servo). Also, please note that the black wire is the ground wire on each servo connection cable.



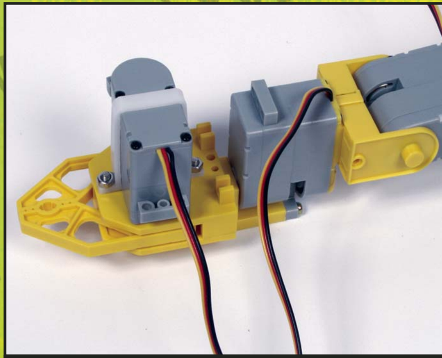
STEP 39. All four legs are plugged into their respective jumper ports.



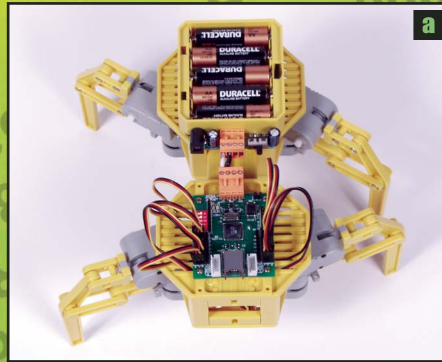
STEP 40. A special serial cable is provided for connecting Quadruped to your PC. This cable is used for testing servo parameters and programming the Quadruped servo controller circuit board.

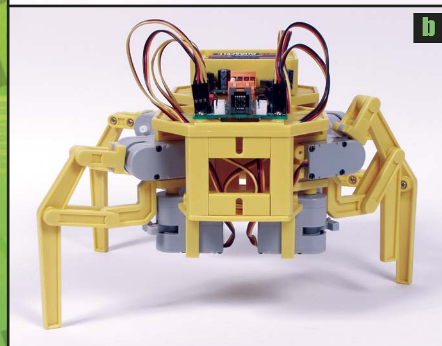


STEP 41. The completed Quadruped. Alkaline batteries were substituted for the rechargeable batteries mandated in the Quadruped User's Manual.

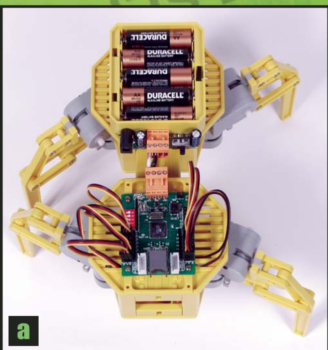


STEP 42. A great hack involves the removal of the gripper mechanism from the JoinMax Smart Arm followed by the installation of this gripper on the Quadruped "head." Just plug the servo connection cables into any available jumper ports and use Mini Servo Explorer for running the gripper.





STEP 43. a: A plan view of Quadruped's first step. A sample forward step with the robot's left legs numbered 1 (the leg attached to the "head") and 3 (the battery box leg) and Quadruped's right legs numbered 2 (the "head" leg) and 4 (the battery box leg). **b:** An elevation view of Quadruped's first step. This is the same sample forward step with leg 1 lifted.



STEP 44. a: A plan view of Quadruped's second step. Leg 1 is lowered and moves forward, leg 2 moves backward, leg 3 moves backward, and leg 4 lifts and moves forward. **b:** An elevation view of Quadruped's second step. This is the same sample forward step with leg 4 lifted.

ABOUT THE AUTHOR

Dave Prochnow is a frequent contributor to *Nuts & Volts* and *SERVO Magazine*, as well as the author of 26 nonfiction books including the mega-hit The Official Robosapien Hacker's Guide (McGraw-Hill, 2006) and the upcoming PSP Hacks, Mods, and Expansions (McGraw-Hill, 2006). You can learn more about this Robosapien book and other robotics/electronics projects at Dave's website: **ww.pco2go.com**

EVENTS CALENDAR

Send updates, new listings, corrections, complaints, and suggestions to: **steve@ncc.com** or FAX **972-404-0269**

Unless you're in India, don't expect to see any robot competitions this month. The Robotix competition in West Bengal, India is the only thing on the calendar. Don't feel like making the trip to India? That's okay. Just wait a while and there will be plenty of events coming up in the next couple of months. One of them is likely to be nearby no matter where you live.

I've been keeping track of robot competitions for over 10 years now, and one interesting trend I've noticed in the last couple of years is the proliferation of international events. In the past, the majority of competitions seemed to be in the US and Japan, with a few in the UK or Canada. Today, I'm tracking events all over the world. To name a few, in no particular order: Poland, Korea, Spain, Argentina, the Dominican Republic, Denmark, the Republic of Singapore, Israel, Slovakia, Germany, Portugal, Australia (many kangaroos), Austria (not so many kangaroos), Estonia, India, Russia, and the Netherlands.

I don't think we have any competitions in Antarctica yet. Are there any *SERVO* readers at Vostok or McMurdo? If so, maybe it's time to plan a robot event. Send me some photos. Or better yet, fly me down and I'll be a judge. I've always wanted to visit Antarctica.

In any case, it's encouraging to see people from so many different countries working together on robotics technology. I look forward to seeing the number of countries included on the robot competition list continue to grow.

For last-minute updates and changes, you can always find the most recent version of the Robot Competition FAQ at Robots.net: **http://robots.net/rcfaq.html**

— R. Steven Rainwater

February

3-5 **Robotix**
IIT Khargpur, West Bengal, India
A national-level competition with events such as Mission to Mars, Water Polo, Topsy Turvy, Distance Tracker, and Match Maker.
http://gymkhana.iitkgp.ac.in/robotix

March

10-11 **AMD Jerry Sanders Creative Design Contest**
University of Illinois at Urbana, Champaign, IL
This year's contest involves a 44' x 44' course in which robots from multiple teams will navigate ramps, overpasses, and teeter-totters in an attempt to collect and dispose of colored balls.
http://dc.cen.uiuc.edu

18-19 **Manitoba Robot Games**
Winnipeg, Manitoba, Canada
Events may include both Japanese and Western style sumo, mini-tractor pull, and Atomic Hockey.
www.scmb.mb.ca

19-23 **APEC Micromouse Contest**
Dallas, TX
One of the best-known micromouse competitions in the United States. Expect to see some very advanced and fast micromouse robots.
www.apec-conf.org

21-23 **Singapore Robotic Games**
Republic of Singapore
Fourteen events including autonomous sumo, RC sumo, legged robot marathon, legged robot obstacle course, several levels of micromouse, wall climbers, pole balancers, and more.
http://guppy.mpe.nus.edu.sg/srg

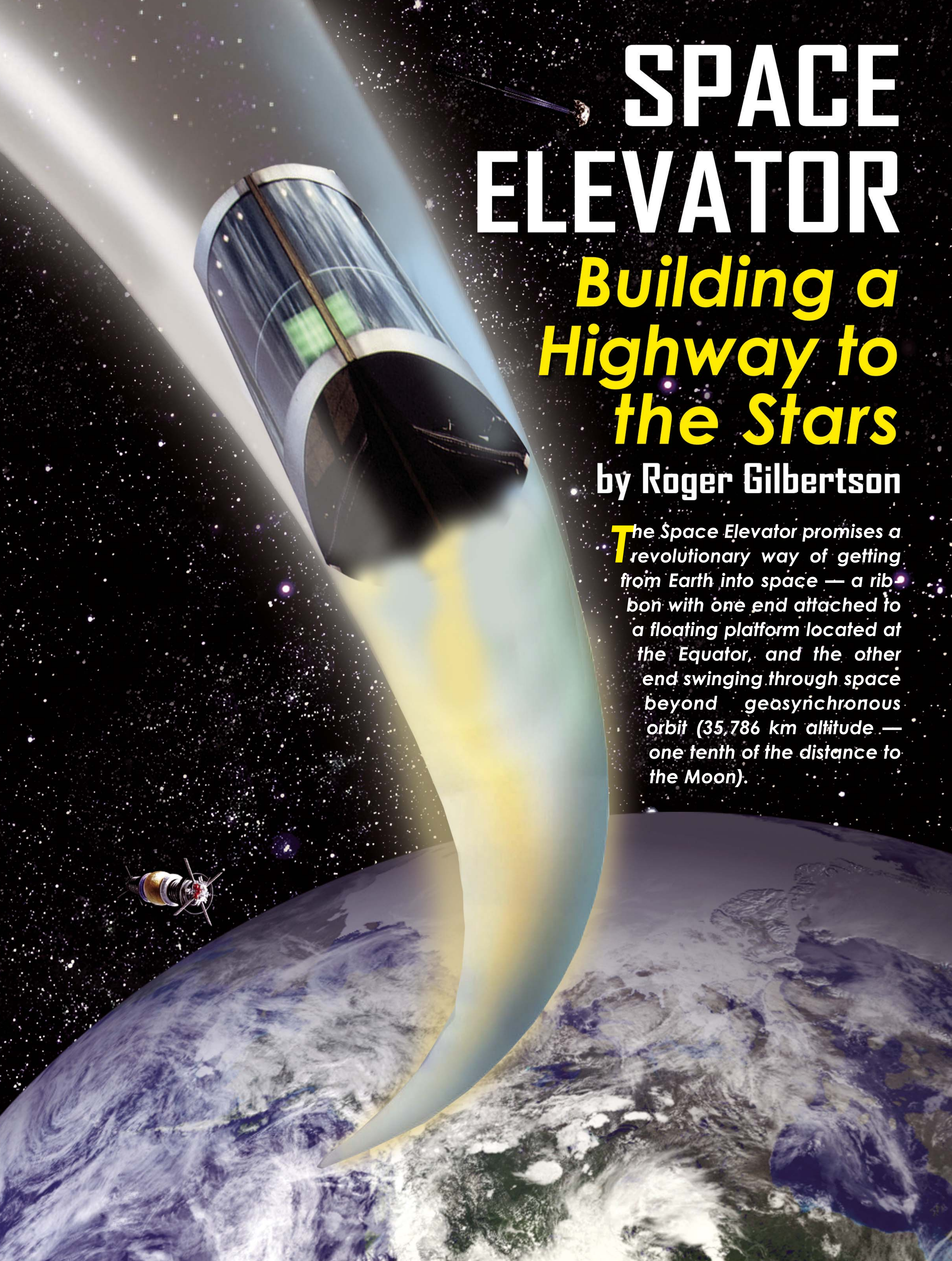
April

8-9 **Trinity College Fire Fighting Home Robot Contest**
Trinity College, Hartford, CT
The well-known championship event for fire fighting robots.
www.trincoll.edu/events/robot

21 **Carnegie Mellon Mobot Races**
CMU, Pittsburgh, PA
The traditional Mobot slalom and MoboJoust events.
www.cs.cmu.edu/~mobot

21-22 **National Robotics Challenge**
Veterans Memorial Coliseum, Marion, OH
In addition to sumo and maze solving events, this student competition includes two unusual ones: a robotic workcell event and a pick-and-place event.
www.nationalroboticschallenge.org

22 **UC Davis Picnic Day MicroMouse Contest**
University of California, Davis campus, CA
Standard micromouse contest.
www.ece.ucdavis.edu/umouse



SPACE ELEVATOR

Building a Highway to the Stars

by Roger Gilbertson

The Space Elevator promises a revolutionary way of getting from Earth into space — a ribbon with one end attached to a floating platform located at the Equator, and the other end swinging through space beyond geosynchronous orbit (35,786 km altitude — one tenth of the distance to the Moon).

The Space Elevator would provide an incredibly low-cost way to ferry satellites, spacecraft, space station components, and people into space using robotic climbers clamped to the ribbon. It would serve as a means for commerce, scientific advancement, and exploration.

The Space Elevator would be the largest and greatest engineering project ever, and an advance perhaps as significant as any achieved by man. By providing cheap and easy access to space, a space elevator would forever transform human possibilities in this arena and open the gates to colonizing the stars.

An Old Vision for a New Future

In July 1960, nearly three years after the flight of the first artificial satellite, Sputnik 1, and nine months before the flight of Yuri Gagarin, the first human in space, the Russian daily newspaper "Pravda" carried a full page article describing a bold new idea for taking people to the stars.

"Calmly, not rushing and fussing, passengers will take their seats in hermetically sealed wagons of such a train. For this is not a cosmic rocket whose launching is calculated in fractions of a second. The electric train gives a last whistle, slowly picks up speed, and darts vertically upward on the web of delicate threads. Then the first layer of clouds is left behind. The speed of movement grows ever more ...

Behind are transparent packs of silver clouds — almost a hundred kilometers separates the train from Earth. The speed grows and grows: indeed the resistance of the atmosphere becomes less. Unblinking stars sparkle in the black velvet sky of the cosmos."

— "To the Cosmos by Electric Train," Komsomolskaya Pravda, July 31, 1960

In his proposal, engineer Yuri Artsutanov identifies one major obstacle. "There is still no material whose durability could bear the gigantic weight of a cable from Earth to the cosmos. The most durable plastics and steels are several times weaker than required."

With the Cold War heating up and the Moon Race just ahead, Artsutanov's vision found little attention outside the Soviet Union.

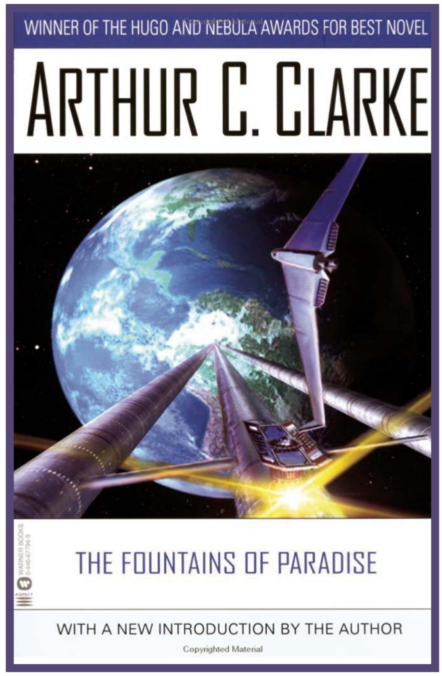


Figure 2. ***The Fountains of Paradise.***

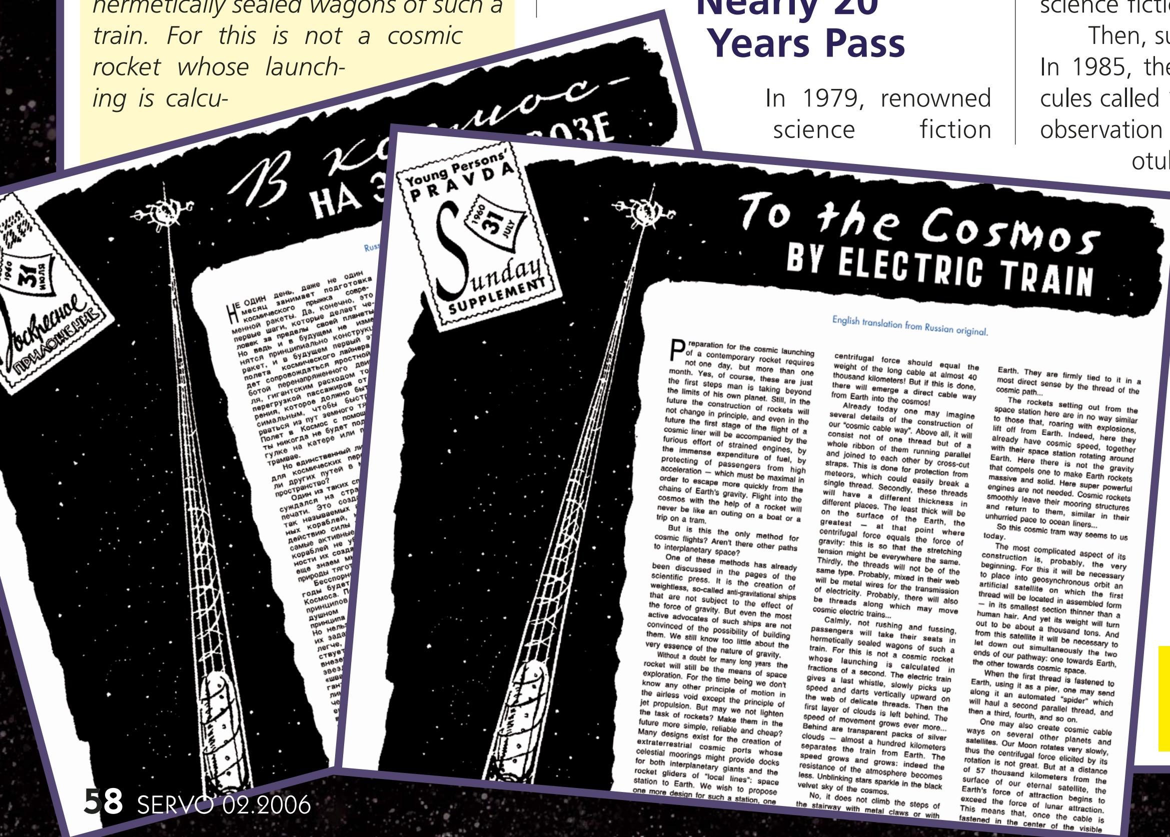
Nearly 20 Years Pass

In 1979, renowned science fiction author Arthur C. Clarke presented the space elevator concept in his novel *The Fountains of Paradise*. In a technical paper published shortly after, Clarke acknowledged Artsutanov's inspiration. Although now exposed to the wider world, the space elevator concept remained firmly in the realm of science fiction.

Then, suddenly everything changed. In 1985, the discovery of carbon molecules called "fullerenes," followed by the observation of super-strong carbon nanotubes, meant that there now existed — at least in the laboratory — a material 10 times lighter and 250 times stronger than steel. Carbon nanotubes suddenly transformed the Space Elevator from a science fiction fantasy to a tantalizing possibility.

A New Vision

In 2000, NASA's Institute for Advanced



Preparation for the cosmic launching of a contemporary rocket requires not one day, but more than one month. Yes, of course, these are just the first steps man is taking beyond the limits of his own planet. Still, in the future the construction of rockets will not change in principle, and even in the future the first stage of the flight of a cosmic liner will be accompanied by the furious effort of strained engines, by the immense expenditure of fuel, by protecting of passengers from high acceleration — which must be maximal in order to escape more quickly from the chains of Earth's gravity. Flight into the cosmos with the help of a rocket will never be like an outing on a boat or a trip on a tram.

But is this the only method for cosmic flights? Aren't there other paths to interplanetary space?

One of these methods has already been discussed in the pages of the scientific press. It is the creation of weightless, so-called anti-gravitational ships that are not subject to the effect of the force of gravity. But even the most active advocates of such ships are not convinced of the possibility of building them. We still know too little about the very essence of the nature of gravity.

Without a doubt for many long years the rocket will still be the means of space exploration. For the time being we don't know any other principle of motion in the airless void except the principle of jet propulsion. But may we not lighten the task of rockets? Make them in the future more simple, reliable and cheap? Many designs exist for the creation of extraterrestrial cosmic ports whose celestial moorings might provide docks for both interplanetary giants and the rocket gliders of "local lines": space station to Earth. We wish to propose one more design for such a station, one

centrifugal force should equal the weight of the long cable at almost 40 thousand kilometers! But if this is done, there will emerge a direct cable way from Earth into the cosmos!

Already today one may imagine several details of the construction of our "cosmic cable way". Above all, it will consist not of one thread but of a whole ribbon of them running parallel and joined to each other by cross-cut straps. This is done for protection from meteors, which could easily break a single thread. Secondly, these threads will have a different thickness in different places. The least thick will be on the surface of the Earth, the greatest — at that point where centrifugal force equals the force of gravity: this is so that the stretching tension might be everywhere the same. Thirdly, the threads will not be of the same type. Probably, mixed in their web will be metal wires for the transmission of electricity. Probably, there will also be threads along which may move cosmic electric trains...

Calmly, not rushing and fussing, passengers will take their seats in hermetically sealed wagons of such a train. For this is not a cosmic rocket whose launching is calculated in fractions of a second. The electric train gives a last whistle, slowly picks up speed and darts vertically upward on the web of delicate threads. Then the first layer of clouds is left behind. The speed of movement grows ever more... Behind are transparent packs of silver clouds — almost a hundred kilometers separates the train from Earth. The speed grows and grows: indeed the resistance of the atmosphere becomes less. Unblinking stars sparkle in the black velvet sky of the cosmos.

No, it does not climb the steps of the stairway with metal claws or with

Earth. They are firmly tied to it in a most direct sense by the thread of the cosmic path...

The rockets setting out from the space station here are in no way similar to those that, roaring with explosions, lift off from Earth. Indeed, here they already have cosmic speed, together with their space station rotating around Earth. Here there is not the gravity that compels one to make Earth rockets massive and solid. Here super powerful engines are not needed. Cosmic rockets smoothly leave their mooring structures and return to them, similar in their unhurried pace to ocean liners...

So this cosmic tram way seems to us today.

The most complicated aspect of its construction is, probably, the very beginning. For this it will be necessary to place into geosynchronous orbit an artificial satellite on which the first thread will be located in assembled form — in its smallest section thinner than a human hair. And yet its weight will turn out to be about a thousand tons. And from this satellite it will be necessary to let down simultaneously the two ends of our pathway: one towards Earth, the other towards cosmic space.

When the first thread is fastened to Earth, using it as a pier, one may send along it an automated "spider" which will haul a second parallel thread, and then a third, fourth, and so on.

One may also create cosmic cable ways on several other planets and satellites. Our Moon rotates very slowly, thus the centrifugal force elicited by its rotation is not great. But at a distance of 57 thousand kilometers from the surface of our eternal satellite, the Earth's force of attraction begins to exceed the force of lunar attraction. This means that, once the cable is fastened in the center of the visible

Figure 1. Excerpt from July 31, 1960 Pravda article by engineer Yuri Artsutanov.

Concepts provided Dr. Brad Edwards with a modest grant to evaluate the prospects for building a space elevator system using carbon nanotube technology.

Edwards concluded "a space elevator can be built in the near future with current technologies combined with a yet-to-be developed carbon nanotube composite."

"It sounds a little far out at first, but with materials science advances such as nanotubes and other new materials, we are reaching the stage where this starts to look like real science, a real advance for space transport."

Basic Elevator Science

In his NASA-funded study, Brad Edwards envisions the Space Elevator.

"Springing out from an anchor point on the Equator, the space elevator cable would rise straight up, passing through geostationary orbit at 36,000 km and continuing for another 64,000 km until it terminates in a 600-ton counterweight. The cable would be held up in a manner similar to that which holds a string taut as a weight tied to it is swung in a circle. The key detail that would make the elevator work would be the fact that its center of gravity would be at the geostationary orbit mark, forcing the entire structure to move in lockstep with Earth's rotation.

Electrically powered elevator cars, which I call climbers, would crawl up the cable, carrying people or cargo. Each car would weigh about 20 tons fully loaded, of which about 13 tons would be payload. These payloads could be in the form of inflatable structures, like those proposed for the International Space Station, with about 900 cubic meters of space, or roughly as much as a five-bedroom house. For passengers, a climber would be like a space-going cruise ship; there would be small sleeping quarters, a tiny kitchen and other amenities, and, of course, windows with some of the most stunning views in the solar system. Ascending at 190 km per hour, the climbers would reach geostationary orbit in about eight days.

The flexible ribbon, just one meter wide and thinner than paper, would be made of carbon-nanotube composite fibers arranged in long strands, cross-braced to evenly redistribute the load if a strand were cut. Space debris that would sever a small round cable would pass through the broader ribbon, creating small holes and a manageable reduction in cable strength, letting it survive impacts from small debris and meteoroids, which would be fairly common ...

Because of the thinness of the ribbon, it would be surprisingly light: the entire 100,000 km length would have a mass of just 800 tons, not counting the counterweight's 600 tons."

— Dr. Brad Edwards, IEEE "Spectrum," August 2005

Figure 3. A climber ascending a ribbon to orbit (illustration by The Space Elevator Visualization Group).

Technical Challenges Abound

Many daunting obstacles confront the engineers of a space elevator system — not just the need for incredibly strong ribbons and high-energy power beaming — but environmental challenges like hurricanes, tornados, lightning, hail, and with increasing altitude, problems with high-speed winds, extreme temperatures, atomic oxygen corrosion, meteorites, satellites, space junk, radiation, tidal forces, harmonic oscillations, and other demons.

People often ask what would happen if the ribbon should break? In short, the length of ribbon above the break flies outward into space, with the portion below fluttering gently downwards, as it has the density of plastic sandwich wrap.

Space junk presents possibly the most challenging of the obstacles, but already NORAD performs extensive tracking of all objects, and objects can be avoided by moving the elevator's anchor platform, and by other

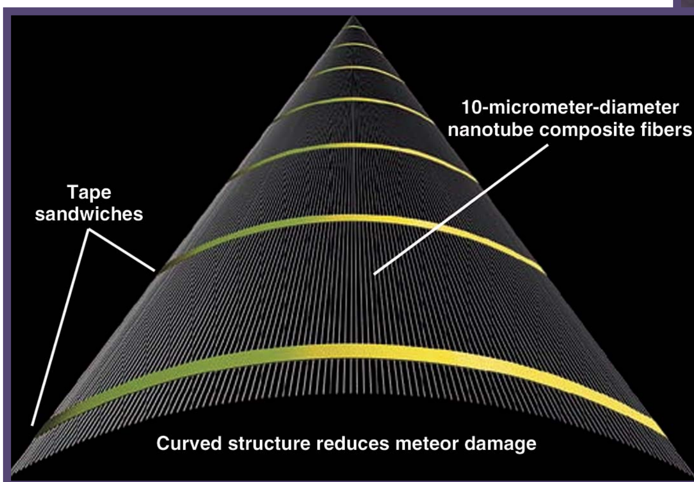


Figure 4. Cross-section of a ribbon.

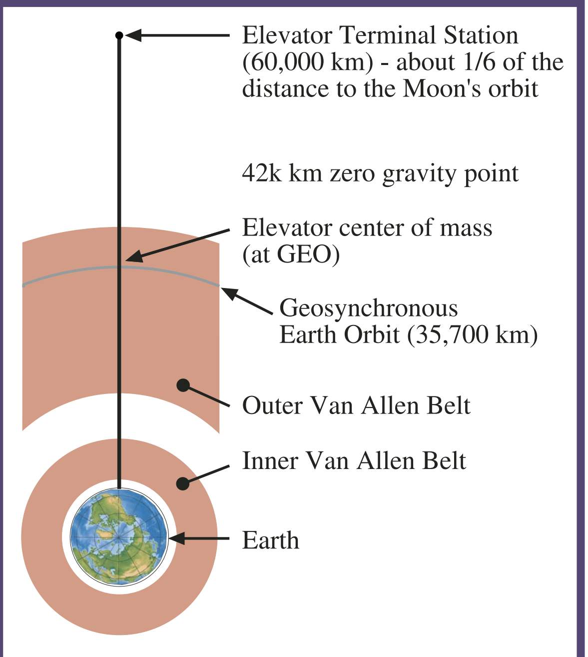


Figure 5. Plan view of a space elevator.

techniques. Locating the platform in a remote area reduces the risk of sabotage.

Edwards concluded the ideal location to anchor the first space elevator would be on the equator, straight south of California, about 2,000 miles west of Ecuador. This region has a very low incidence of hurricanes and other extreme weather, and minimal air and ship traffic.

Financial Challenges

So who might build and pay for a space elevator? Private companies like those that built the English Channel tunnel? Government agencies like NASA? Or a combination of both as with the transcontinental railroad of the 1860s? And who should benefit? The builders? The governments? All the people of Earth?

An elevator would open space to even poorer countries, and provide the cheap access to space needed for expanding space tourism and establishing extraterrestrial settlements.

Compared to NASA's proposed $100 billion budget for returning to the Moon, a working Space Elevator for $20 billion seems a bargain, and certainly in the realm of private enterprise.

But Why?

As science fiction writer Robert Heinlein observed, once you reach Earth orbit, you're halfway to anywhere. To its proponents, the space elevator provides inexpensive, high-volume access to the vast material and energy resources of space. To those who look to create a prosperous future for all humanity, greatly expanded human presence in space holds the highest potential of answering some of the significant problems facing humans on Earth:

- The increasing need for clean and sustainable energy sources.
- The growth of human population beyond the Earth's carrying capacity.
- Increasing pollution, rising CO_2 levels, global warming, polar ice melting, and changing weather patterns.
- Threats of global disaster from cosmic collisions with comets or asteroids.

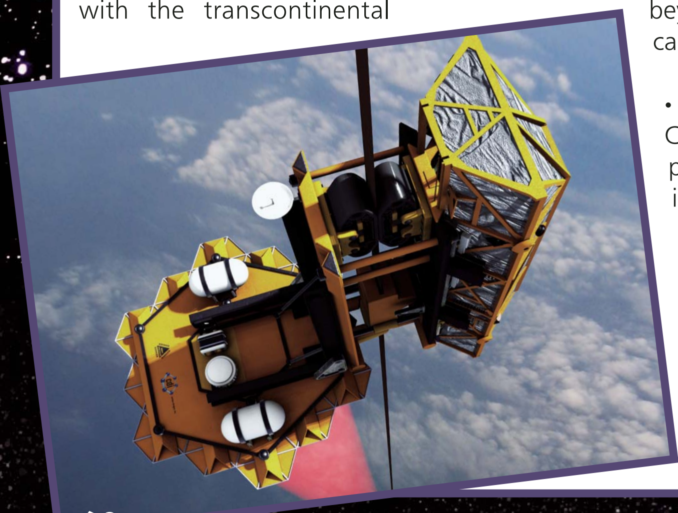


Figure 6. Laser energy, beamed from the Earth, powers a climber as it ascends a ribbon to orbit (illustration by The Space Elevator Visualization Group).

The material resources available in the inner solar system dwarf all of those so far used by humanity. The space elevator provides a new hope to achieve an ever more prosperous and sustainable existence — both on Earth and beyond.

> *"In order to save the human race, we must develop the technologies that will allow us to live and work on other places in the Solar System ... SINGLE PLANET SPECIES DO NOT LAST and we have no idea how much time we have."*
>
> **— John Young, former Gemini, Apollo, and Space Shuttle astronaut, October 2002**

The Contrarian View

Skeptics justifiably confront the space elevator concept with numerous questions:

Will current material science provide adequate technology to create the "magic ribbon?"

Is there a sufficient market for such a system? For example, the year 2002 saw less than 40 commercial space payloads. There is no guarantee that greatly reduced cost to orbit would result in an immediate increase in demand. The non-successes of the great private satellite ventures of the late 1990s cause optimistic space business projections to be met with much skepticism.

What are the governmental considerations, red tape, and legal ramifications? International law requires that objects in space be registered with their host country, and that the host be responsible for any damage. Numerous federal agencies already make life difficult for space entrepreneurs and established businesses alike.

A space elevator system that reaches across so many domains —

water, surface, air, and space — also crosses numerous legal and regulatory regions, creating lots of new areas for lawyers to explore.

Event 1.0 — Robotic Society of America's Space Elevator Challenges

In 2002, inspired by Brad Edward's space elevator vision, I created an event for the San Francisco Robotics Society of America's annual Robot Games. Called "The Space Elevator Ribbon Climbing Robot Competition," the event tasked robot builders with the challenge of making a device weighing less than 1 kilogram (2.2 lbs) capable of scaling a thin polyethylene ribbon 30 mm (1+3/16 inch) wide.

Climbers could carry all their power on board (as with batteries), but machines that received some or all of their power from a "beamed" source — such as high intensity flashlights located at the launch platform — would receive a much higher scoring factor.

The December 2002 event saw but one successful entry. Nick Donaldson assembeled his climber, S.E.R. (for Space Elevator Robot), entirely from LEGO blocks and operated it by a LEGO MindStorms RCX controller. It completed a 2.4 meter (eight foot) climb in 37 seconds, and earned Nick a place in the history books as the first Space Elevator ribbon climbing robot.

The second running of the event — in March of 2004 — saw a great increase in the number of participants, in part due to the $1,000 prize offered by the Institute for Scientific Research (ISR), a government sponsored research lab that was home to Brad Edwards and his continuing work on space elevator development.

Ten teams signed up, and five entrants completed the 3.56 meter (11 feet, 8 inch) climb, with two operating under beamed energy. Jack Buffington's elegant and fast climber — ironically named Racing Slug — took first place, the big cash prize, and a spot in history as the first ever fully beam powered Space Elevator ribbon climbing robot.

Figure 7. Donaldson and his climber "S.E.R." built from LEGO MindStorms.

Figure 8. Buffington with Racing Slug — the first beam powered climbing robot success.

Event 2.0 — NASA Fuels the Fires

The scope and scale of space elevator competitions was soon to change dramatically. On October 23, 2005, NASA stepped into the game by supplying over $100,000 in prize money for a new event called the Space Elevator Games — the premier event of NASA's new X-Prize-styled series of Centennial Challenges.

Held at NASA's Ames Research Center in Mountain View, CA, the three days of grueling competition and friendly shoulder-to-shoulder innovation, were designed to address the technical as well as "social engineering" issues of the space elevator. These two engineering competitions are intended to generate interest and excitement in academia, the space enthusiast community, and the general public.

The Beam Power Challenge tasks designers with building an unmanned machine, weighing 50 to 100 pounds, capable of pulling itself up a four-inch wide, 200-foot long ribbon suspended from a crane, and powered only

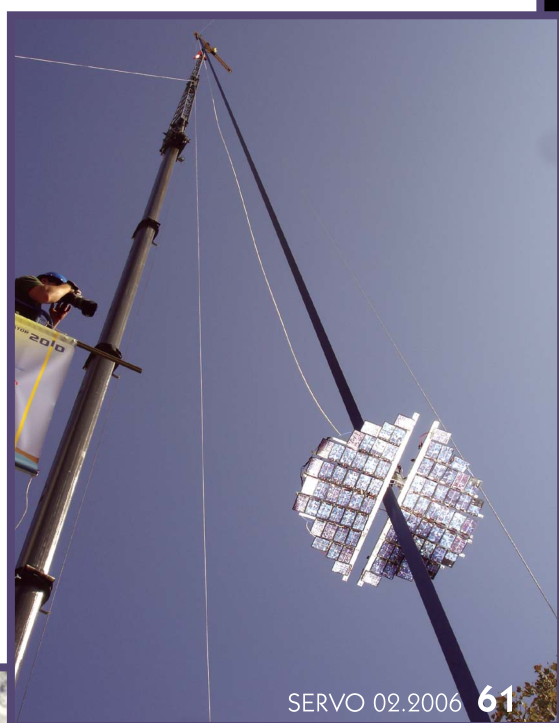


Figure 9. The University of British Columbia entry makes its way skyward during NASA's first Centennial Challenge competition held in October 2005.

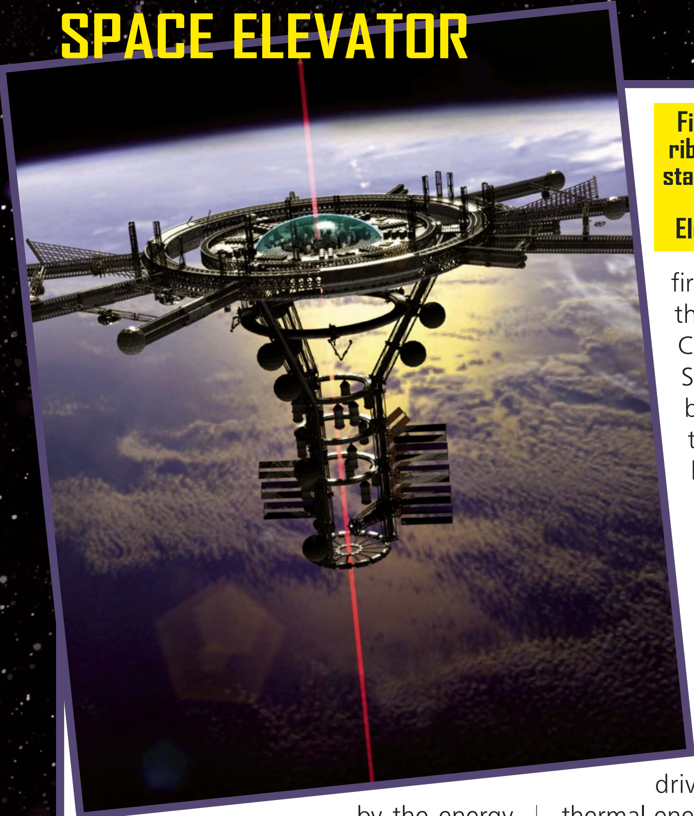


Figure 10. The space elevator ribbon passes through a space station at geosynchronous orbit (illustration by The Space Elevator Visualization Group).

by the energy beamed up from a 10,000 watt xenon searchlight.

Seven teams vied for the $50,000 first prize, five from across the US and two from Canada. The University of Saskatchewan team — led by Edwin Zhang — reached the highest altitude under beamed power, about one third of the full distance. At that point, their one-square meter array of space-grade solar cells yielded insufficient energy to continue. Other entrants used various solar array schemes and even Stirling engines driven by the searchlight's thermal energy, but no team managed to complete the challenge.

In the second event — the Tether Challenge — four teams offered their best formulation for an ultra-light, ultra-strong ribbon material. In one-on-one tug-of-wars, each of the

LINKS

For more information about the Centennial Challenges program
http://centennialchallenges.nasa.gov

For more information on the NASA Space Elevator Centennial Challenge competitions
www.elevator2010.org

For more information on the Spaceward Foundation
www.spaceward.org

For more information on the RoboGames Space Elevator Ribbon Climbing events
http://robogames.net

entrants were tested to their breaking points. In the final round, the strongest contender, fielded by Centaurus Aerospace of Logan, UT, yielded at 1,260 pounds, giving way to the "house ribbon" which broke at just over 1,300 pounds of force, and leaving the $50,000 first prize unclaimed.

As in the first year of the DARPA Grand Challenge for autonomous vehicles, the Space Elevator Challenges set high goals, and seek to inspire innovation, as well as public attention.

"This year's unclaimed prize money will be added to next year's, and that alone should boost the number of entrants," said Brant Sponberg, manager for NASA's Centennial Challenges program. "This is an exciting start. We don't need to try to pick and choose which approaches might be successful. We simply invite all new ideas to come and give it a shot, and then reward the best," he said.

With the 2006 climber and tether challenge first prizes set at $150,000 each, and with smaller awards for second and third places, over 20 teams have already signed up. Indeed, some of next year's hopefuls were seen in attendance, learning from this year's events and taking notes.

Will we be seeing your robotic climber entering next year's competition?

Conclusion

The space elevator could represent the first step in a new space age. Whether driven by hubris and greed, by a passion for solving great challenges, or by a deep concern for humanity, what we choose to build reveals the most essential aspects of what makes us human — the desire to create, to turn our dreams into reality, and to build a more promising future for our descendants.

The fabric of history is woven on the weft of great adventures and the bold experiments of those who've said "it can be done."

A space elevator would exceed the scope of all past engineering, exploration, and space projects, and encompasses a grandly historic vision, comparable to the greatest of past human adventures:

- The ocean crossing journeys of the Polynesians.
- China's great armadas of exploration nearly 100 years before Columbus.
- The voyages of Columbus, Magellan, Cook, and many others.
- The building of the transcontinental railroad and the Panama Canal.
- The Apollo missions to the Moon.

Dreamers, acting on their dreams, create the future. Just as the transcontinental railroads replaced wagon trains, so could space elevators replace rockets and in the process blast the space frontier wide open to all. **SV**

FOR YOUR INFO ...

For an in-depth look at Space Elevator technology, check out L. Paul Verhage's Near Space column in the **September 05** issue of *Nuts & Volts* (**www.nutsvolts.com**).

ABOUT THE AUTHOR

In college, Roger G. Gilbertson studied engineering, robotics, and the walking patterns of living creatures. In 1987, he co-founded Mondo-tronics, Inc., to explore the commercial applications of Shape Memory Alloy wires, and in 1995 launched **RobotStore.com** — the Internet's first commercial robotics site. Mondo-tronics continues to innovative new products for students, educators, hobbyists, and experimenters using shape memory alloys and other innovative technologies. Roger lives and works in Marin County, CA, where robots may be found vacuuming the floors, but don't yet stand a chance at cleaning the closets.

by Dave Calkins

2005 ROBOT SOCCER Championships

Part 1 — RoboCup Osaka

There are lots of robot sports out there — combat robots, Tetsujin, sumo robots, Lego Mindstorms — but the ultimate challenge in robotic sports is Robot Soccer. In combat robots, you design and build and machine from the ground up and drive it, but there are no sensors or artificial intelligence (AI). In sumo bots, you just find anything that moves and push it. Robo-Ones — as cool as they are — are still remote-controlled. The ultimate challenge for the robot builder is robot soccer. Build a fast robot, with full AI, that can localize not only against a single opponent, but multiple opponents, a ball, and two opposing goals.

Two Organizations

Soccer robots are akin to combat robots in that there are many classes and two leagues. Both started around the same time; RoboCup and FIRA (Federation of International Robot Soccer Association) both promote the same goals — multiple entry-level soccer leagues with a difficult but not impossible task, with the ultimate goal of fielding humanoid robots the same size as humans who can independently compete against actual humans in a regulation soccer competition.

"As a competitor, FIRA feels more like the Sundance film festival whereas RoboCup is more like the Oscars," said Jacky Baltes, Associate Professor for Computer Science at the University of Manitoba, Winnipeg, Canada. Jacky leads the humanoid events at both RoboCup and FIRA. "RoboCup has more glitz and glamour, but FIRA does not charge you more than $1,000 to participate and show your robot."

RoboCup

The shining goal of RoboCup is a date, not a machine. The goal is that by 2050, RoboCup can field a team of independently programmed life-sized humanoid robots that can beat the FIFA WorldCup championship team. It sounds easy. But once you consider the real difficulties that individual members of a soccer team must face — finding the ball, knowing friend from foe, coordinating with the other members of your team, only kicking towards the opponent's goal (own goals being all too common in today's human played games), and knowing when to play and when to listen to the referees.

The first RoboCup games were held in Nagoya, Japan in July 1997. A total of 330 teams from 31 countries with 2,000 participants competed in the soccer, rescue, and junior competitions setting a new record number of participants in the history of RoboCup. Much like combat events, the competitors are always helping each other and sharing code, tools, and even helping each other on-site. But once a match starts, it's all about winning!

> “There are lots of robot sports out there — Combat robots, Tetsujin, sumo robots, Lego Mindstorms — but the ultimate challenge in robotic sports is Robot Soccer.”


Photo 1

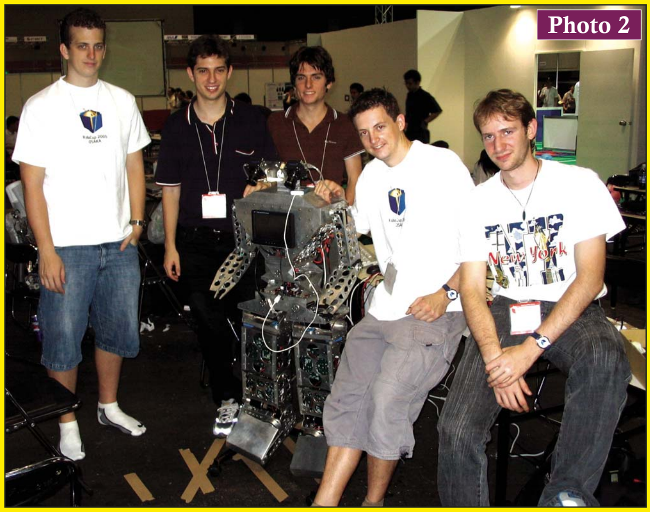


Photo 2

Photo 3


Photo 4

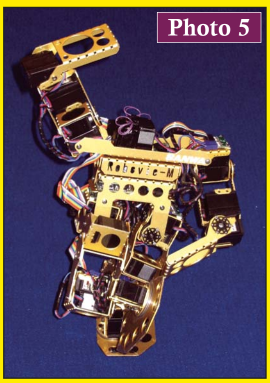


Photo 5


Photo 6

Photo 7

PHOTO 1. Dogs and cats fight it out in an Aibo league match.

PHOTO 2. Not all humanoid robots are as small as cats, as the German team proves.

PHOTO 3. High-schoolers show off their abilities in the Junior League.

PHOTO 4. A familiar sight at all robot events ...

PHOTO 5. The VisiON prototype — “Robovie-M” shows its poise.

PHOTO 6. Humanoids duke it out in the penalty kick phase.

PHOTO 7. Celebrating a score!

2005 Robot Soccer Championships

Professor Baltes summarizes the challenges accurately, “People from all over the world try and solve some really, really hard problems in a competitive environment. Even though there is a competition, all researchers — and especially students — realize that they are part of something bigger — *‘The quest to understand human reasoning better’* — and, thus, to have a great impact on our society.”

Varying Events

RoboCup is divided into three major divisions, 10 leagues, and 15 competitons (Yeah, it’s even harder to understand than the free-kick rules in human soccer ...). RoboCup Soccer Division consists of the small-size league (seven inches square); middle-size league (20 inches square and weighing almost as much as a human); four-legged league (Aibos — probably the easiest entry point and certainly the most popular, both with teams and with audiences); humanoid league, which includes four different events (most robots are about 15” tall like Robo-Ones, but they can be as big as four feet tall); and the Simulation League, which doesn’t involve actual robots but perfects the software agents which control bots (2D, 3D, and Coach competitions are included).

The RoboCup Rescue Division is a more practical event. One of the most promising areas of robotics — not for profit but for truly aiding humanity — is rescuer robotics. When earthquakes, tsunamis, fires, terrorism, or other events collapse buildings and cities, robots will play a greater role in helping find survivors while they’re still alive. The RoboCup Rescue project diverges from the soccer goal but still focuses on robotics. The rescue robot league has actual robots moving through a simulated collapsed building looking for survivors, while the rescue simulation league is computer simulated.

RoboCup Junior is a project-oriented set of challenges for pre-collegiate students. This gives them an insight into robotics, and gets them involved in basic electronics, hardware, and software. Unlike many high-school robotic events, RoboCup Junior has three independent challenges with consistent rules that students can pick from: Soccer Challenge, Rescue Challenge, and the Dance Challenge.

ROBOCUP 2005 CHAMPIONS

Best Humanoid “Louis Vuitton Humanoid Cup”
Team Osaka — Japan

2 on 2 Kid-size Humanoid League
1. Team Osaka — Japan
2. NimbRo — Germany
3. Team Hajime — Japan

Penalty Kick Humanoid League
1. Team Osaka — Japan

Technical Challenge Humanoid League
1. Team Osaka — Japan
2. NimbRo — Germany

2D Simulation League
1. Brainstormers 2D — Germany
2. Wright Eagle 2005 — China
3. Tokyo Tech SFC — Japan

3D Simulation League
1. ARIA — Iran
2. Brainstormers 3D — Germany
3. ZJUBase — China/Caspian and Iran (tie)

Coach Simulation League
1. UT Austin Villa — USA
2. Aria — Iran
3. Kasra — Iran

Small-size Robot League
1. FU-Fighters — Germany
2. Cornell Big Red — USA
3. Field Rangers — Singapore

Middle-size Robot League
1. EIGEN Keio Univ — Japan
2. FU-Fighters — Germany
3. Philips RoboCup Team — The Netherlands

Four-legged Robot League
1. German Team — Germany
2. NuBots — Australia
3. rUNSWift — Australia

Rescue Robot League
1. Toin Pelican — Japan
2. ROSQUE — Korea
3. Casualty — Australia

Agent Competition Rescue Simulation
1. Impossibles — Iran
2. Caspian — Iran
3. Kshitij — India

VisiOn Winners — First Step to a Butler Bot?

The biggest winner of RoboCup 2005 was Team Osaka and their robot, VisiOn. In the history of RoboCup, no humanoid robots have ever completed all of the humanoid trials, much less place first in them all. In fact, prior to this event, no humanoid robot had ever managed to finish the technical challenge. And yet, this year, VisiOn managed to complete the technical challenge and finish first in all humanoid events (2 on 2, Penalty, and Technical Challenge).

Wouldn’t it be great to have eyes in the back of your head? The obvious point being to see in both directions. One of the critical innovations of Team Osaka’s robot is its head — not its brain, but its physical head. Instead of trying to replicate human eyes that look forward, Team Osaka devised a completely new (and much improved) way of seeing. A single camera mounted above the neck looks directly up. Above the camera is a mirrored cone opposite the camera, which gives the robot a 360° view of its surroundings. So with a single eye, it sees everything in front, to the left, to the right, and behind itself (just like your mom could when you were trying to sneak a cookie, but with much more wiring ...).

Watching the Team Osaka robot play against the other humanoid is like comparing a middle-schooler soccer student with David Beckham. Typical humanoids have a problem standing up, much less successfully defending a goal. The VisiOn robot practically waltzed around its opponents (most of whom fell down while merely trying to

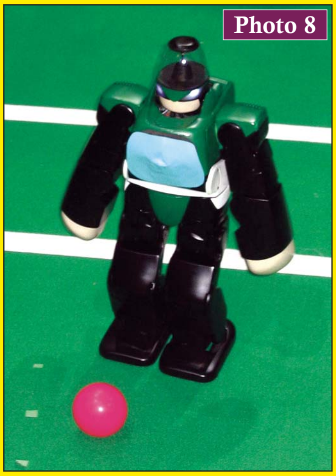


Photo 8

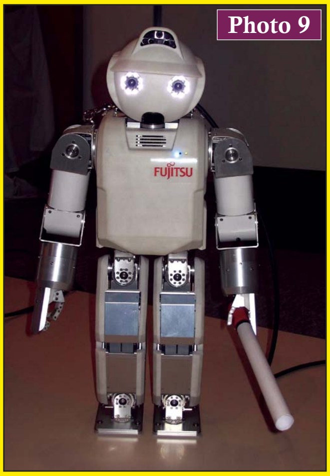


Photo 9

Photo 10

Photo 11

Photo 12

Photo 13

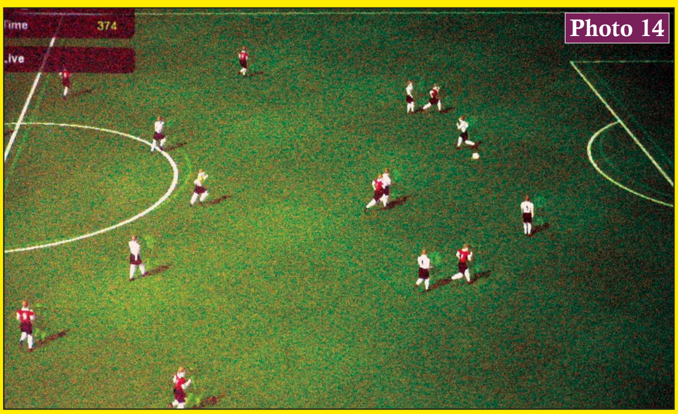


Photo 14

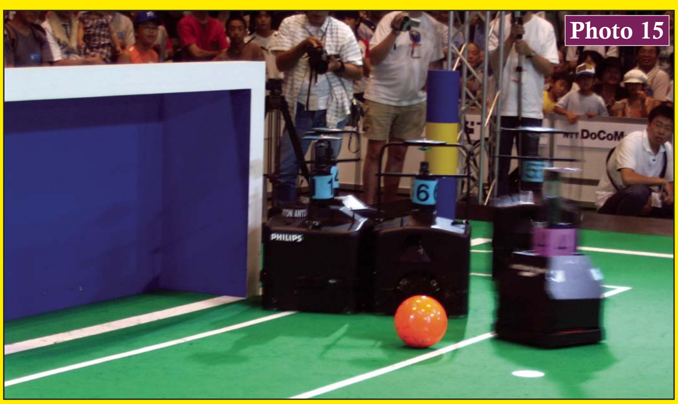


Photo 15

PHOTO 8. The grand champion VisiON robot lines up on a ball.

PHOTO 9. Fujitsu's newest humanoid is almost two feet tall and carries his own light saber.

PHOTO 10. Small league robots are small, but quick!

PHOTO 11. Team Osaka shows off their baby.

PHOTO 12. Almost as glorious as the ComBots Cup, the Louis Vuitton Humanoid Cup was won by Team Osaka two years running.

PHOTO 13. Preparations for the middle-league rounds.

PHOTO 14. 3D simulation league games are even more exciting than human soccer games.

PHOTO 15. He shoots, he scores!

2005 Robot Soccer Championships

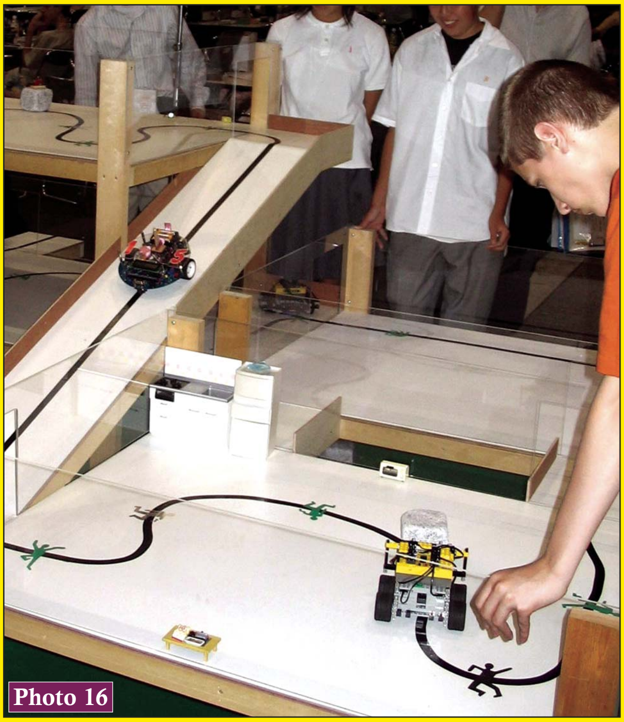


Photo 16


Photo 17


Photo 19


Photo 18

Photo 20


Photo 21

PHOTO 16. Junior league line-following robots do their stuff.

PHOTO 17. While no physical robots are involved, the simulation-league perfects the AI behind the bots.

PHOTO 18. Rescue robots have many hands that need helping.

PHOTO 19. A rescue-league robot navigates through the maze.

PHOTO 20. 2D simulation league.

PHOTO 21. The Japanese Team calibrates their small-league robots.

approach the ball).

"The VisiOn robot has some impressive moves — my favorite one is the goalkeeper that dives for the ball and then gets up again, and the way they try and steal the ball from other players," said Baltes. "Other researchers — me included — try to learn how to control a humanoid with low quality motors and sensors, since I believe that human intelligence is amazing because of its ability to compensate for noise and uncertainty (as introduced by these motors and sensors) in the world. Entertainment-type humanoids are available now and it is a matter of economics. The VisiOn and similar robots would sell like hot cakes if they would cost less than $10,000. However, their cost is more like $50,000. Bigger, 'useful' humanoid robots that can do most of the household chores are still much further away."

"... all researchers — and especially students — realize that they are part of something bigger — *'The quest to understand human reasoning better'* — and, thus, to have a great impact on our society."

As is typical at robot competitions, beds were not part of the program. "We didn't sleep for the past two days. We won't sleep tonight to do the best for tomorrow," said one Team Osaka competitor. Their commitment was rewarded with the Louis Vuitton Cup for best overall humanoid.

Soccer Events in 2006

If you're looking to either compete in RoboCup, or to just watch a match, there are lots of events coming up. The US Open will be held in Atlanta, GA April 22nd-25th, the WorldCup will be in Bremen, Germany June 14-20th, and ROBOlympics will once again host robot soccer June 16-18th in San Francisco, CA — along with the *SERVO*-sponsored 2006 Tetsujin event, robot combat, and many other events.

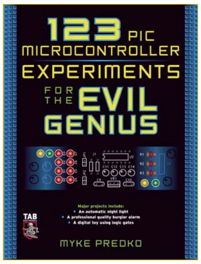
Next month, we'll cover the FIRA 2005 Championship, which was held in Singapore. SV

Mind Candy For Today's Roboticist

The SERVO Store

123 PIC Microcontroller Experiments for the Evil Genius

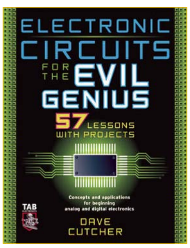
by Myke Predko



Few books take advantage of all the work done by Microchip. *123 PIC Microcontroller Experiments for the Evil Genius* uses the best parts, and does not become dependent on one tool type or version, to accommodate the widest audience possible. Building on the success of *123 Robotics Experiments for the Evil Genius*, as well as the unbelievable sales history of *Programming and Customizing the PIC Microcontroller*, this book will combine the format of the evil genius title with the following of the microcontroller audience for a sure-fire hit. **$24.95**

Electronic Circuits for the Evil Genius

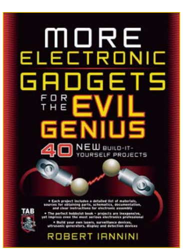
by Dave Cutcher



Cutcher's 57 lessons build on each other and add up to projects that are fun and practical. The reader gains valuable experience in circuit construction and design and in learning to test, modify, and observe results. Bonus website **www.books.mcgraw-hill com/authors/cutcher** provides animations, answers to worksheet problems, links to other resources, WAV files to be used as frequency generators, and freeware to apply your PC as an oscilloscope. **$24.95**

MORE Electronic Gadgets for the Evil Genius

by Robert E. Iannini



This much anticipated follow-up to the wildly popular cult classic *Electronic Gadgets for the Evil Genius* gives basement experimenters 40 all-new projects to tinker with. Following the tried-and-true Evil Genius Series format, each project includes a detailed list of materials, sources for parts, schematics, documentation, and lots of clear, well-illustrated instructions for easy assembly. Readers will also get a quick briefing on mathematical theory and a simple explanation of operation. **$24.95**

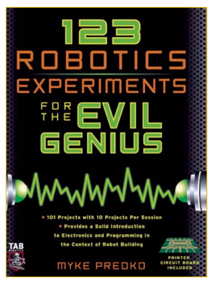
Mechatronics for the Evil Genius

by Newton C. Braga

If you're fascinated by electronics and mechanics, this hands-on tour of the junction where they meet will bring you hours of fun and learning. Noted electronics author Newton Braga's *Mechatronics for the Evil Genius* guides you step-by-step through 25 complete, intriguing, yet inexpensive projects developed especially for this book. You will build your own mechanical race car, combat robot, ionic motor, mechatronic head, light beam remote control, and 20 other entertaining learning projects that take you to the heart of mechatronics. **$24.95**

123 Robotics Experiments for the Evil Genius

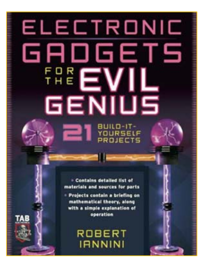
by Myke Predko



If you enjoy tinkering in your workshop and have a fascination for robotics, you'll have hours of fun working through the 123 experiments found in this innovative project book. More than just an enjoyable way to spend time, these exciting experiments also provide a solid grounding in robotics, electronics, and programming. Each experiment builds on the skills acquired in those before it so you develop a hands-on, nuts-and-bolts understanding of robotics — from the ground up. **$25.00**

Electronic Gadgets for the Evil Genius

by Robert Iannini

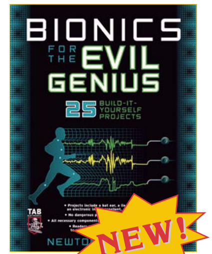


The do-it-yourself hobbyist market — particularly in the area of electronics — is hotter than ever. This book gives the "evil genius" loads of projects to delve into, from an ultrasonic microphone to a body heat detector, all the way to a *Star Wars* Light Saber. This book makes creating these devices fun, inexpensive, and easy. **$24.95**

We accept VISA, MC, AMEX, and DISCOVER
Prices do not include shipping and may be subject to change.

Bionics for the Evil Genius

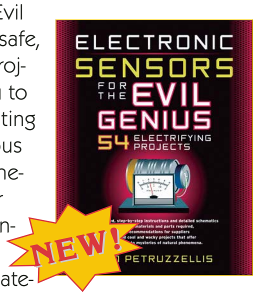
by Newton C. Braga



Step into the future (or the past, if you have a touch of Dr. Frankenstein in your soul) with these 25 incredibly cool bionic experiments! Demonstrating how life forms can be enhanced, combined, manipulated, and measured with electronic and mechanical components, these inexpensive projects from internationally renowned electronics guru Newton Braga provide hours of fun and learning. Totally safe, *Bionics for the Evil Genius* guides you step-by-step through 25 complete, intriguing, and low-cost projects developed especially for this book — including an electric fish, a bat ear, a lie detector, an electronic nerve stimulator, a panic generator, and 20 other exciting bioelectric/mechanical projects! **$24.95**

Electronic Sensors for the Evil Genius — 54 Electrifying Projects

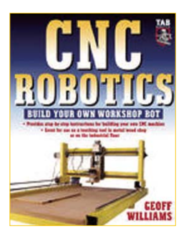
by Thomas Petruzzellis



Nature meets the Evil Genius via 54 fun, safe, and inexpensive projects that allow you to explore the fascinating and often mysterious world of natural phenomena using your own home-built sensors. Each project includes a list of materials, sources for parts, schematics, and lots of clear, well-illustrated instructions. Projects include rain detector, air pressure sensor, cloud chamber, lightning detector, electronic gas sniffer, seismograph, radiation detector, and much more! **$24.95**

CNC Robotics

by Geoff Williams

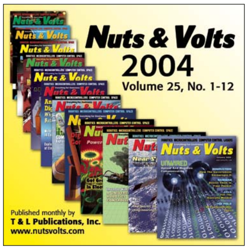


Now, for the first time, you can get complete directions for building a CNC workshop bot for a total cost of around $1,500.00. *CNC Robotics* gives you step-by-step, illustrated directions for designing, constructing, and testing a fully functional CNC robot that saves you 80 percent of the price of an off-the-shelf bot and can be customized to suit your purposes exactly, because you designed it. **$34.95**

To order call 1-800-783-4624 or go to our website at www.servomagazine.com

Mind Candy For Today's Roboticist

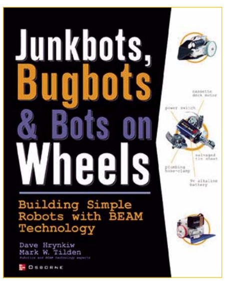
Nuts & Volts CD-Rom



Here's some good news for *Nuts & Volts* readers! Starting with the January 2004 issue of *Nuts & Volts*, all of the issues through the 2004 calendar year are now available on a CD that can be searched, printed, and easily stored. This CD includes all of Volume 25, issues 1-12, for a total of 12 issues. The CD-Rom is PC and Mac compatible. It requires Adobe Acrobat Reader version 6 or above. Adobe Acrobat Reader version 7 is included on the disc. **$29.95**

JunkBots, Bugbots, and Bots on Wheels

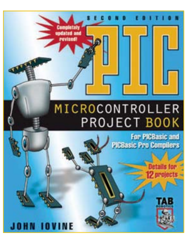
by Dave Hrynkiw / Mark W. Tilden



From the publishers of *BattleBots: The Official Guide* comes this do-it-yourself guide to BEAM (Biology, Electronics, Aesthetics, Mechanics) robots. They're cheap, simple, and can be built by beginners in just a few hours, with help from this expert guide complete with full-color photos. Get ready for some dumpster-diving! Get step-by-step instructions from the Junkbot masters for creating simple and fun self-guiding robots safely and easily. **$24.95**

PIC Microcontroller Project Book

by John Iovine



The PIC microcontroller is enormously popular both in the US and abroad. The first edition of this book was a tremendous success because of that. However, many users of the PIC are now comfortable paying the $250.00 price for the Professional version of the PIC Basic. This new edition is fully updated and revised to include detailed directions on using both versions of the microcontroller, with no-nonsense recommendations on which one serves better in different situations. **$29.95**

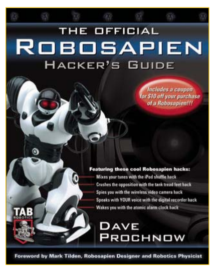
Check out our online bookstore at www.servomagazine.com for a complete listing of all the books that are available.

SERVO CD-Rom

Are you ready for some good news? Starting with the first *SERVO Magazine* issue — November 2003 — all of the issues through the 2004 calendar year are now available on a CD that can be searched, printed, and easily stored. This CD includes all of Volume 1, issues 11-12 and Volume 2, issues 1-12, for a total of 14 issues. The CD-Rom is PC and Mac compatible. It requires Adobe Acrobat Reader version 6 or above. Adobe Acrobat Reader version 7 is included on the disc. **$29.95**

The Official Robosapien Hacker's Guide

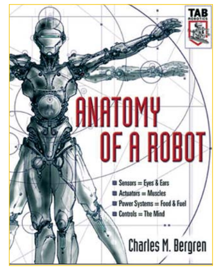
by Dave Prochnow



The Robosapien robot was one of the most popular hobbyist gifts of the 2004 holiday season, selling approximately 1.5 million units at major retail outlets. The brief manual accompanying the robot covered only basic movements and maneuvers — the robot's real power and potential remain undiscovered by most owners — until now! This is the official Robosapien guide — endorsed by WowWee (the manufacturer) and Mark Tilden (the designer). This timely book covers possible design additions, programming possibilities, and "hacks" not found any place else. **$24.95**

Anatomy of a Robot

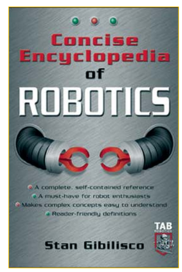
by Charles Bergren



This work looks under the hood of all robotic projects, stimulating teachers, students, and hobbyists to learn more about the gamut of areas associated with control systems and robotics. It offers a unique presentation in providing both theory and philosophy in a technical yet entertaining way. Reading *Anatomy of a Robot* is like having a robot on the operating room table. Crack open the pages, and you'll be able to dissect a robot from head to toe — or from project management and design to motors and power systems. **$29.95**

Concise Encyclopedia of Robotics

by Stan Gibilisco



This handy collection of straightforward, to-the-point definitions is exactly what robotics and artificial intelligence hobbyists need to get and stay up to speed with all new terms that have recently emerged in robotics and artificial intelligence. Written by an award-winning electronics author, the *Concise Encyclopedia of Robotics* delivers 400 up-to-date, easy-to-read definitions that make even complex concepts understandable. Over 150 illustrations make the information accessible at a glance and extensive cross-referencing and a comprehensive bibliography facilitate further research. **$19.95**

$20 each 15% off if you buy two or more.

From HomoSapien to RoboSapien

Before R2D2 there was R1D1

SERVO Magazine T-Shirts

Get your very own limited-edition *SERVO Magazine* T-shirt. Shirts come in sizes S, M, L, and XL, and are available in either black or white. All shirts are 100% pre shrunk cotton.

```
// castling bonuses
B8 castleRates[]={-40,-35,-30,0,5};

//center weighting array to make pieces prefer
//the center of the board during the rating routine
B8 center[]={0,0,1,2,3,3,2,1,0,0};

//directions: orthogonal, diagonal, and left/right
from orthogonal for knight moves
B8 directions[]={-1,1,-10,10,-11,-9,11,9,10,-10,1,-
1};

//direction pointers for each piece (only really for
bishop rook and queen
B8 dirFrom[]={0,0,0,4,0,0};
B8 dirTo[]={0,0,0,8,4,8};

//Good moves from the current search are stored in
this array
//so we can recognize them while searching and make
sure they are tested first
```

LESSONS FROM THE LABORATORY

A bi-monthly column just for kids!

Electronic Stopwatch for Races

Guest Hosted by Marc Helfman

James is generously allowing me to do a guest spot in his column this month, but I trust he'll be back next time with another interesting topic. In his excellent article in the October, 2004 issue, he talked about the importance of finding the right gear ratio when designing a drag racer. Other factors that can affect the speed of your dragster are wheel size, mechanical configuration, weight, and freshness of your batteries. Anyone who's tried to optimize a racer can tell you that a lot of experimentation is involved in getting these variables just right. Sometimes it's obvious that a particular gear or wheel selection speeds up your 'bot, but this isn't always the case. So how can you tell if you've really made a difference?

One way would be to use a stopwatch and hope your reflexes are fast enough to press the button at the exact instant your racer starts and once again when it crosses the finish line. But I think you'll find this method won't produce the accuracy you're looking for. Luckily, the RCX and Robolab have built in features that solve this problem handily.

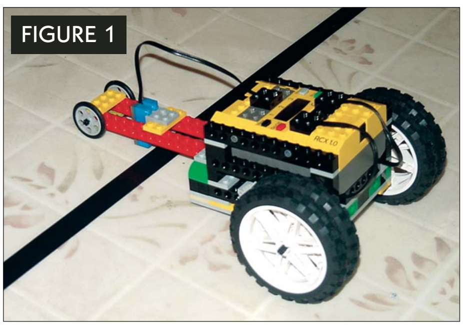
By way of background, we need to talk about two Robolab constructs you may not have worked with before: timers and containers. Lurking in Level 4 of the modifiers sub-palette are red, blue, and yellow icons with tiny alarm clocks on them; these are the three timers that come with Robolab. They work as electronic stopwatches that you can use in your programs to time how long it takes for something to happen. As you'll see later, when we talk about the actual program, there's another icon that you'll need to know about: the "zero timer" command (found in the reset sub-palette).

Many Robolab users never get around to learning about containers (also known as variables), but ask anyone who writes software for a living and they'll tell you they couldn't get along without them. Imagine you have three empty jars sitting on a table, each labeled with a different color: red, blue, and yellow. You write down the ages (say 14, 11, and 12) of three of your friends (whose names just happen to be Red, Blue, and Yellow) on separate pieces of paper, and put each slip into the appropriate jar. A week later, you can't remember Blue's age so you reach into the jar and pull out the slip of paper that reminds you that he's 11 years old.

Containers in Robolab work exactly the same way. You keep track of the container you're using by referring to its color and you store or retrieve the actual number you're interested in by accessing its value. Take a look at the timer icons in the modifier's sub-palette once again. On the bottom left, you'll see the red, blue, and yellow timer labels and to the right you'll see the icons that will contain the values for these timers. When dealing with containers, its easy to become confused between the label on the container and the actual value stored inside it, so hopefully this explanation will help you keep things straight.

Enough with the theory! Let's get to the fun part — writing the program and putting it to use to make a winning drag racer. The photo in Figure 1 shows the drag racer from James' article with its light sensor positioned just ahead of the starting line and poised to race across the finish line 10 feet away. The program we

FIGURE 1



are about to write will act like a built-in stopwatch and display the amount of time your racer takes to get to the finish line.

Referring to Figure 2, we see that the program starts with a five second delay. This is so your hand won't be in contact with the racer once it takes off. Next, we reset the red timer to zero and turn on the motors — the race is on! As we hurtle towards the finish line, the light sensor starts looking for the strip of black tape.

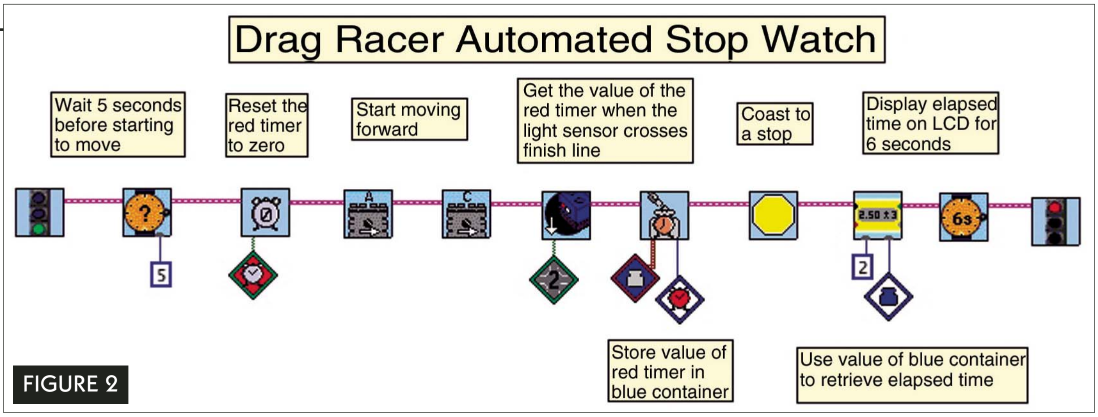


FIGURE 2

Once the racer crosses the finish line, the value of the red timer is stored in a container. I've chosen a blue container just to have a little variety. If you look in the container sub-palette, you'll find two icons with the alarm clocks on them. The one we want is on the right, three rows from the bottom. We've chosen it because it will give a more precise value for our stopwatch, in hundredths of a second.

Now that we've stored the timer value, we allow our bot to coast to a stop and display the elapsed time on the RCX's LCD display. Notice that we've strung in the "value of blue timer" icon to the "set display" icon (which is found in the RCX communications sub-palette) to retrieve our timer value and that we've also strung in a numeric constant of "2" so that we'll get the right decimal point position for displaying hundredths of a second.

Finally, we use a six second "wait for" to keep the timer value displayed long enough for us to walk over and take a look at the LCD.

To test out the program, I raced the bot on a 10 foot course and got a readout of 2.85 seconds; if you do the math, that translates to 3.51 feet per second. Wondering how fast this is in miles per hour? Just multiply the speed we just got by .682 — in my case a whopping 2.4 miles per hour. "Gentlemen, start your engines!" **SV**

by James Antonakos

THE ASSEMBLY LINE

Uno and the Rise of the Phoenix

Ah, the joy of troubleshooting. One might almost look forward to the troubleshooting portion of a project because it means the device has been built. It just does not work yet. This is the case with Uno, whose stepper motor driver IC (the ULN2003 Darlington driver) made some smoke signals during the first test of the completed system. Like the mythical Phoenix, Uno went out in a blaze. But can Uno come back, as the Phoenix does, and be reborn from its ashes?

My first step in troubleshooting this problem involved finding another copy of the datasheet for the ULN2003. I wanted to re-examine the Darlington transistor configuration and have a pinout reference. Next, I unscrewed the stepper motor driver board and looked for something obvious, such as a solder bridge, or two wire-wrap pins bent and touching each other. No luck there, there are no visible signs of trouble with the wiring. That does not mean the driver IC is wired correctly.

So, next, I checked the wiring diagrams that I used during construction. I traced the five colored wires from the stepper motor to the driver IC. The white, red, green, and brown wires were connected to four outputs on the driver IC. So far so good, as these four wires control the coils inside the stepper motor.

This left only the black wire from the stepper motor, which is common to all four coils and must be pulled up to +12V. This is where I noticed something strange. On my schematic and wiring diagram, I had the black wire connected to pin 9 of the ULN2003, as it should be. However, I actually wired it, along with another black wire connected to +12V, to pin 8 of the ULN2003.

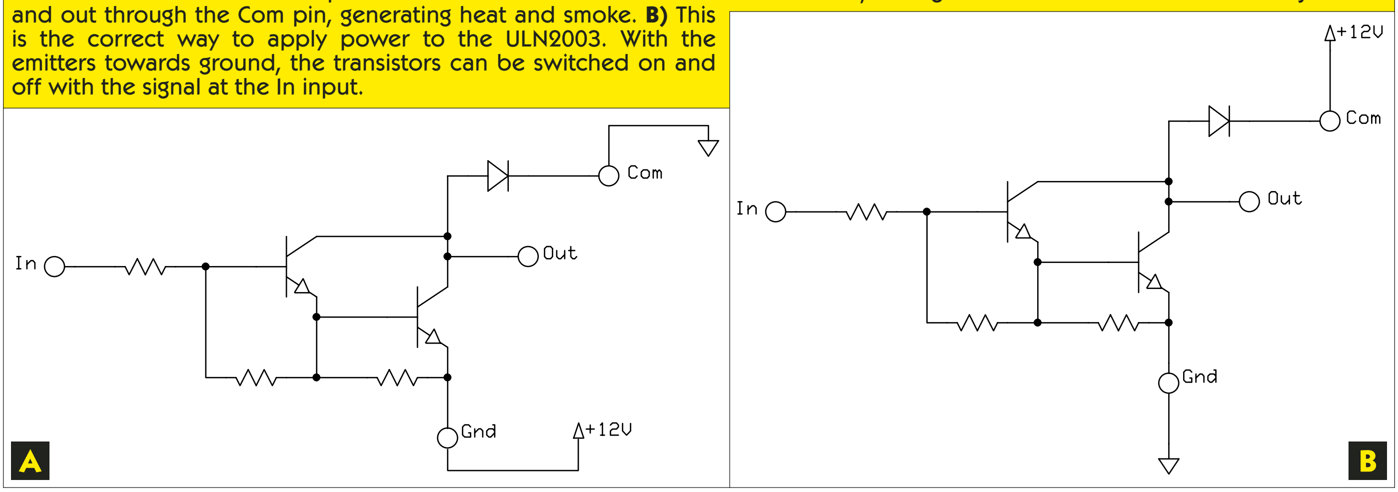
Oops. To make matters worse, the ground wire that was supposed to go to pin 8 was instead wired to pin 9. These two wiring errors were very bad and directly led to the smoky results. Take a look at Figure 1 to see why.

I had now identified three wires to disconnect and re-connect to the proper locations. I soldered the Black wire from the stepper motor directly to the output of the on/off switch. As long as I was making repairs, I thought I would bypass the wire-wrap connector and feed to the motor directly.

I then swapped the two wire-wrap wires that provide +12V and ground to the stepper motor driver board. I also added a second wire-wrap wire to the ground connection, to provide a better path for return current from the ULN2003.

Not being a gambling man, I also

Figure 1. Schematic of the Darlington driver within the ULN2003. **A)** My wiring errors cause +12V and ground to be swapped on the ULN2003's Com and Gnd pins. This allowed lots of current to flow into Gnd, through each transistor's base-collector junction and out through the Com pin, generating heat and smoke. **B)** This is the correct way to apply power to the ULN2003. With the emitters towards ground, the transistors can be switched on and off with the signal at the In input.



replaced the old ULN2003 chip with a new one, in case something melted in the original chip and more trouble lay in wait for me. With my fingers crossed once again, I flipped on Uno's power switch.

And ... no smoke! The rear wheels did their expected forward-reverse cycle, and then the stepper motor did a quarter rotation in each direction, just like the test program tells it to. It seemed too good to be true, so I let Uno run for a short time to see if any smoke reappeared. There was none, the ULN2003 stayed cool to the touch. Say hello to Uno: Phoenix edition.

Now, for those of you thinking "that was too easy," let me both agree and disagree. I agree because it was easy to troubleshoot. I found a wiring error within two minutes of checking the wires. Touchdown!

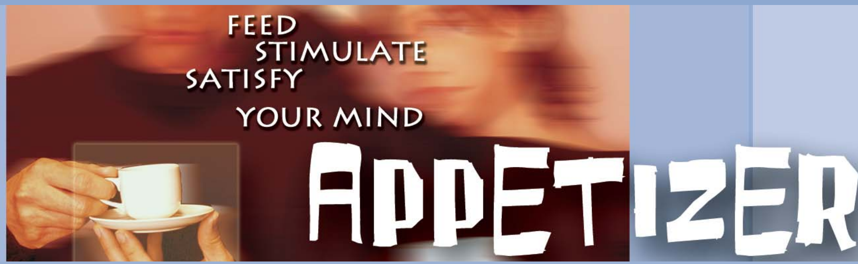
I disagree because, well, it should have been easy. After all, the circuit worked on the breadboard. The hardware and software designs worked. So, the only place an error could have been introduced was when the hardware was moved from the breadboard to the circuit board on Uno's chassis. If I had never breadboarded the design and tested the hardware before connecting it all up, the troubleshooting would have been much more difficult. And to top it all off, where was the mistake? It was in the power wiring, one of the first things I check when troubleshooting any circuit.

So, now that I am all proud of myself and everything, I can attend to the remaining mechanical issues associated with Uno's rear wheel assembly and front wheel. After that, I'll teach Uno a few tricks and present him in all his glory next time. **SV**

ABOUT THE AUTHOR

James Antonakos is a Professor in the Departments of Electrical Engineering Technology and Computer Studies at Broome Community College, with over 27 years of experience designing digital and analog circuitry and developing software. He is also the author of numerous textbooks on microprocessors, programming, and microcomputer systems. You may reach him at antonakos_j@sunybroome.edu or visit his website at ***www.sunybroome.edu/~antonakos_j***

Robotics Showcase



The Problem With R2D2

by Erik Hagman

This last October, I was exhibiting with my brother at RoboNexus 2005 and a gentleman from a webzine, whose name I forget (sorry), walks up. He asks me if I could define what a robot is for his publication and that he was asking other exhibitors at the show the same thing. For me, this was a very important and perplexing question, which was evident in more than a couple places in the show.

Just What Exactly is a Robot?

The thought always brings images of R2D2/C3P0 of Star Wars' fame or massive Bolos from Laumer's science fiction. But even these two sets of images are ambiguous to a certain degree since, as machines, they have little in common. The two Star Wars inventions appear to be very general-purpose while the Bolos are very specific in design, that of warfare. They all have very advanced brains and are autonomous in operation, apparently interacting with the human or the distinctly bipedal world of Lucas' creation very successfully.

At this point, you might think that this would lead to something. Over the last couple months, our office gets calls from someone who found us on the Internet or via the phonebook from TV, or the movie industry which wants robot parts for a TV commercial, an episode of a series, or a movie — all but one were looking for industrial arms. This is where the term robot starts to get messed up.

We need more of a standard — not necessarily in appearance, but in functionality. A standard base to build on with common, interchangeable component design that can be built by many manufacturers to a common interface standard.

Robotics is often compared to the PC industry in its infancy, built in garages and hoping for that big innovation that will make robots a household item. The personal computer was relatively easy to define — a personal computer for home/business use on which you could do whatever computing you wanted. It was successful probably because of this — it was pervasive and generic. A tabula rasa (blank slate) of common computing, if you like.

It brings me to PC Bots — or PC based robots. PCs — like Via Technologies' Mini-ITX boards — are low power, small form factor, and affordable. Companies like Phidgets are making great USB interfaces and components to augment the huge PC peripheral market. And you probably have heard of White Box Robotics and their 914s which — if and when launched — will be one of the first PC robot bases on the scene.

I imagine many are saying microcontrollers are the way to go, and I would agree for the "toy" types of robot kits, and many levels of education. Microcontrollers are highly efficient in many ways. But aiming for general interoperability requires raising the bar so that problems we didn't envision can be addressed on the platform. This generally means excess processing power and potentially unlimited interfacing capability, which is pretty close to what a modern PC can give you. Microcontrollers will still find places in the interface to the PC in some circumstances or by specific design needs (i.e., low level functionality; say a sleep mode with the main PC "brain" off).

Price vs. Power

The robots I mentioned in my introduction all have one thing in common: a relatively good level of intelligence. Some call it artificial intelligence (AI), but I am not sure what is so artificial about it. This is what I studied in university and this term itself has some problems with definition but, that would be much longer with more philosophy than seems constructive. AI, to get to the point, needs a lot of horsepower. You can write very small bits of code that are pretty 'smart' and efficient, but a general intelligence needs substantial resources be that speed, memory, storage, or sensors. Today, you can buy a microcontroller board for about $100 or a Via motherboard for about $150. The performance difference is staggering — orders of magnitude.

Tools

Whether you are a Linux fan, a Java coder, or a .Net disciple, there are FREE tools for you to use. Just about everything in the Linux and Java world is free. Microsoft — a huge supporter of PC robotics — provides the Express .NET tools for

free now, and they are very powerful and easy to use. What about the OS — Linux is still free (depending where you get it); Windows is going to be available in many new lightweight and lower cost forms often bundled with a new PC robot base, so this should be a non-issue in many ways if you choose to go the Microsoft way.

Our PC Projects

We always have a few projects on the go. Some custom development, others internal, and others just for fun. We have a five-foot, four-inch robot for medical telepresence in the lab, which was an easy decision to base on PC components. Addressing the need for robotics in university classrooms, we are launching a new PC base system called Avatar which supports the Microsoft robotics initiative. But the most enlightening project from the PC bot perspective is R2D2.

Yes, you read that right — R2D2. Almost four years ago, we were contracted to build the brain and interfaces for a "prototypical" R2D2 for an astute collector who also works in the special-effects industry. The mechanical design is almost entirely aluminum, with a very complex and expensive exacting components. This robot is fully functional ... well almost. It has the ability to transition from two to three legs and back again, can communicate in distinctly R2 fashion, has a fully functional head, can project a video image on a wall (video holography is still difficult), can play music, DVDs, function as a PC, transmit video of its actions, deploy antennae and periscope and — of course — it is mobile.

The problem was that we started with microcontrollers and as the design creeped, we needed more and more and we still couldn't achieve all the design goals and interface requirements (video touchscreen). To make a very long story short, we ended up redesigning the entire brain around a Via motherboard. Not only did it save time, it ends up being much cheaper, significantly more powerful, and infinitely more flexible.

With PC bots, I think that we may be on the verge of a new age of robotics development which will usher in the beginnings of the robots we all dream about.

PS: I deliberately didn't define robot. If you want something to build on, I would redefine the question to what is an autonomous mobile robot? **SV**

AUTHOR BIO

*Erik Hagman is the president of Rogue Robotics (**www.roguerobotics.com**). He studied artificial intelligence, Psychology, and Engineering in University. It is his job to tinker with robot designs, design custom robots for customers, and experiment with intelligent machines. In his spare time, he continually modifies his automated home with the goal of one day giving it more intelligence, too.*

Then and NOW

SURGICAL ROBOTS COME OF AGE

by Tom Carroll

For years, the only "serious" factory-built commercial robots were those we've seen for decades in factories around the world. They've long been in space and in the sea as teleoperated and autonomous explorers of places we cannot go as humans. They have now entered the battlefield and the air above it. All of us who have been experimenting with robots for years have finally seen commercially available robots enter the home to vacuum and scrub our floors courtesy of iRobot, and entertain us as robotic pets, courtesy of Sony and others.

When we think of robots in the medical field, we used to envision electro-mechanical anthropomorphic arms and legs used to replace missing human appendages. When asked if we would want to be operated on by a robot, a little shiver runs down our spine. Hearing a surgeon mumble "oops" as he's operating on us is bad enough, but to have an autonomous or even teleoperated machine delving into our innermost parts is a bit unnerving to us all.

Seeing our own creations do unplanned things on our workbenches makes us wonder just what a surgical robot might do if the power failed or had a voltage spike, or a bad bit in a memory chip played havoc on the robot's operation. In questioning a dozen or so people about robotics surgery, I found a much smaller percentage of robot experimenters who would volunteer to undergo surgery assisted by a robot than those of the "general public" who seem to trust that new technology that they don't understand. Very interesting. Make way for the robot doctors.

Figure 1. Zeus Robot Arms.

Back in the late 1980s, a modified industrial robot was used to assist in surgery at Long Beach Memorial Hospital in Long Beach, CA. As a resident of Long Beach for over 30 years, I had to go and see that robot in person. A PUMA 200 robot was used to perform a computer tomographic (CT) guided stereotactic brain surgery. With the patient's head held in a fixed and known position, the robot's computer could use 3-D data derived from CT scans and x-rays to precisely guide the robots end effector to a particular part of the brain. At the time, the PUMA was a small and very popular industrial robot used in the assembly of small parts weighing in the category of a few ounces to a couple of pounds. The main obstacle to the continued use of this industrial robot was its inherent inaccuracy, so only large tumors close to the surface of the skull could be operated on. A NASA-developed calibration technique increased the accuracy of the PUMA but it was still a decade before the FDA approved a true surgical robot. I never got to see the LB Memorial robot "operate" on an actual person but I did get to examine the robot and see numerous videos and TV accounts of the robot's success.

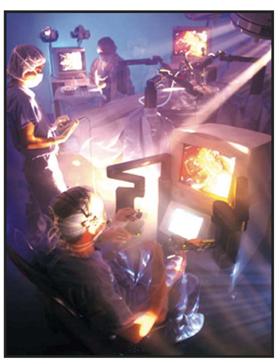


Figure 2. Zeus by Computer Motion.

We're over five years into the new millennium and there are close to a thousand surgical robots in operation around the world. Unlike the LB Memorial robot that only assisted the brain surgeon by inserting a guide tube to the tumor area for the surgeon, the new batch of surgical assistants are actually equipped with a myriad of surgeon's "tools," such as scalpels, forceps, scissors, and other familiar operating room instruments. The "stepping stones" to robotic surgery started about 20 years ago when surgeons began practicing what came to be known as minimally invasive surgery, or MIS. They used instruments called endoscopes or laparoscopes that could be inserted in the patient's body through small incisions to see and

operate. Patient recovery time was dramatically cut as blood loss and pain was minimal.

MIS did encounter significant technical drawbacks. The surgeon had to operate using a standard TV monitor instead of looking at his or her hands. Instead of a natural 3-D depth-of-field, the view was flat and it was hard to see near and far objects. The fixed-wrist instruments limited his/her manual dexterity. As a result, this type of MIS turned out to be suitable for a narrow range of surgical procedures such as simple appendix and gall bladder removal procedures. The new surgical robotic systems have changed all that.

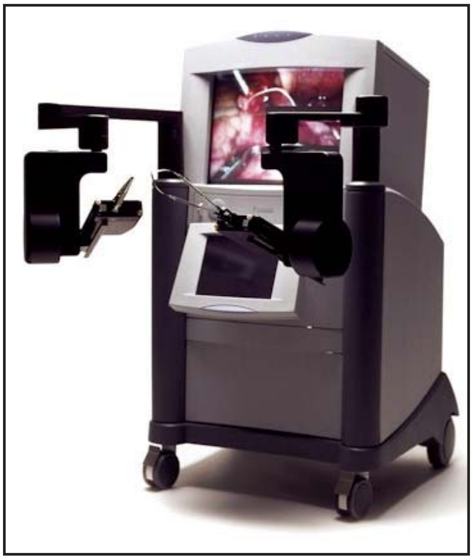


Figure 3. Zeus Console.

Figure 4. da Vinci Surgical System.

There are two robotic surgical systems that have received FDA clearance in the United States. The first is the da Vinci Surgical System, made by Intuitive Surgical, Inc., of Sunnyvale, CA. It has been cleared by the FDA to perform surgery under the direction of a surgeon. The ZEUS Robotic Surgical System, made by Computer Motion, Inc., of Goleta, CA, has also been cleared by the FDA to assist surgeons. Notice the key word: *assist*. Neither of these two systems can perform surgery on its own; a surgeon must be "in the loop."

ZEUS

Computer Motion was founded in 1989 and wanted to revolutionize surgical practices to improve patient lives. They specialized in medical robotics and manufactured systems for the Intelligent Operating Room of the future. Their first product was AESOP, a robotic system used for holding an endoscopic camera in laparoscopic surgery.

The system uses three robotic arms

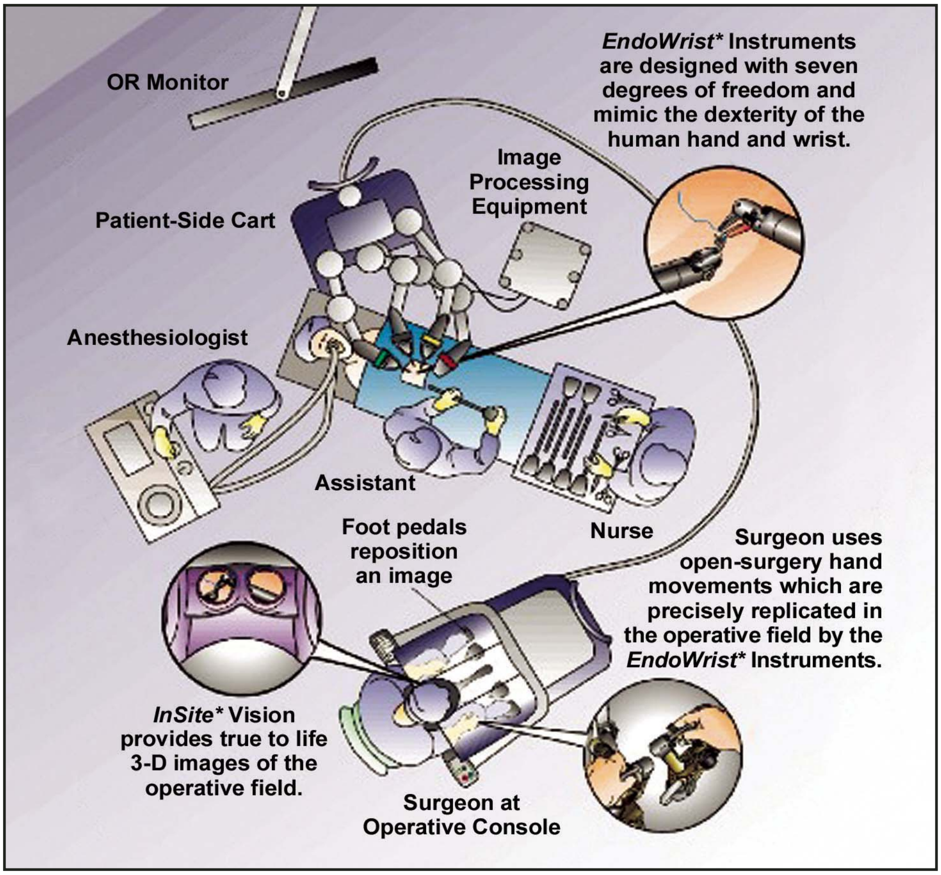


Figure 5. da Vinci Surgical System in a General Procedure Setting.

that are mounted on the operating table. One robotic arm is called the Automated Endoscopic System for Optimal Positioning Robotic System (AESOP). AESOP is a robot used to hold and position the endoscope (camera). The system was like a third arm of the surgeon and was operated by foot pedals. The FDA approved the Aesop 1000 system in 1993 and it became the first surgical robotic device certified by the FDA. The Aesop 2000 released in 1996 used voice control with newer models adding more arms and degrees-of-freedom, finally resulting in the Zeus system in 2001 (Figure 3).

Just four days before the 9/11 terrorist attacks in 2001, US and French doctors and medical scientists performed a transoceanic operation completely by remote control using Computer Motion's Zeus System. The operating surgeons were in New York and the patient was in Strasbourg, France. Through a high-quality telecommunications circuit, the doctors in New York guided the movements of the three-armed Zeus robot in Strasbourg — about 3,870 miles away, and removed the gallbladder of a 68-year-old woman. Two surgeons were closely watching the patient in Strasbourg in order to stop the robotic system if there were complications. "One had his foot on an emergency pedal," said Steven Butner, a professor of electrical and computer engineering at the University of California, Santa Barbara, "If he took his foot off the pedal, the robot would freeze and he could take over," Butner said. The operation took 54 minutes and the patient recovered without complications and went home two days later, according to the medical team.

Figure 6. da Vinci Stereo Camera Probe.

Figure 7. Hand Control.

Just A Tool, Not A "Smart" Machine

"It's a gallbladder operation today, but in the future it could be any kind of medical procedure," said team member Michael Gagner, chief of the Department of Laparoscopic Surgery at Mount Sinai Medical Center in New York. "As the technology evolves and becomes available, and wiring (long distance, high speed data) is more widespread, it will be useful for tele-mentoring, teaching, and performing rare surgery that requires different expertise," Gagner added. "A smaller city could have the help of an expert surgeon just by being connected. The surgical robot is not a 'smart' machine, guided by artificial intelligence, but only a tool. All judgments and movements must be done by surgeons," Gagner said. "The robots were doing exactly what we told them to do from here in New York."

The da Vinci Surgical System

In 1995, the other competitor — Intuitive Surgical — entered the field of robotic surgery. It was formed on robotic surgery technology developed at Stanford Research Institute (SRI) International, long a leader in robotics. Intuitive Surgical collaborated with IBM, MIT, Olympus, Johnson & Johnson, and Medtronic, Inc., and developed the da Vinci System, with loose ties for the name to the famous inventor's old "robot" designs (Figure 4). Some surgeons have jokingly called the $1.5 million robot system "the most expensive pair of scissors ever made." If you are facing the tricky surgery required for prostate cancer or need a heart valve repaired, the da Vinci Surgical System may be "just what the doctor ordered."

Figure 5 shows the operating room layout of the da Vinci Surgical System. The da Vinci offers true-to-life 3-D vision with a bright, crisp, high-resolution image view of the surgical field using dual, three-chip cameras — one of the main selling points of this system. "The da Vinci System provides visualization of the target anatomy

unlike that ever experienced, with natural depth-of-field, enhanced contrast and magnification for more accurate tissue identification and tissue layer differentiation."

Notice the dual lenses and light sources in the stereoscopic probe in Figure 6. The video feedback can allow the surgeon to imagine he's actually inside the patient, doing the surgery in person. One of the drawbacks that surgeons have mentioned is virtual "feel" of the tissues and the interacting instruments. Though there is some tactile feedback, a surgeon normally uses sensations to feel and palpitate an organ or tumor in his surgical procedures. The length of the instruments as compared with standard instruments has also given surgeons a bit of hesitancy.

Operating the da Vinci System, a surgeon sits at a console a dozen or so feet away from the operating table and manipulates the robot's surgical instruments. The robot has three robotic arms attached to a free-standing cart. One arm holds a camera (endoscope) that has been placed in the patient through small incisions. The surgeon operates the other two arms and end-effectors by inserting his fingers into rings at the control console. The robot's arms use a technology called EndoWrist — flexible wrists that surgeons can bend and twist like human wrists.

Figure 7 shows the surgeon's hand control. The surgeon uses hand movements and foot pedals to control the camera, adjust the focus, and position the robotic arms. The da Vinci has a camera system that magnifies the surgical field up to 15 times. Another surgeon stands by the patient adjusting the camera and instruments if needed.

There are four robotic arms, each with seven degrees of freedom and 90 degrees of articulation. The da Vinci System is designed to allow surgeons to operate while seated, a position that is not only more comfortable, but also offers reduced surgeon fatigue. The da Vinci System is a computer-enhanced system that uses a computer between the surgeon's hands and the tips of his instruments to replicate his movements in real time. It cannot be programmed to make movements on its own.

Despite the lack of true autonomy, the term "robotic surgery" is still used to refer to this "computer-enhanced" technology. The present da Vinci System cannot be used over long distances as with the Zeus System, though some design changes could certainly make it so.

Fail-safe "safety" is very important for such a system. To start the procedure, the surgeon's head must be placed in the viewer. Otherwise, the system will lock and remain motionless until it detects the presence of the surgeon's head once again. There is a backup battery that allows the system to run for 20 minutes when power fails, giving the hospital enough time to reestablish power or the surgeon to terminate the surgery. Even the end-effector instrument tips are coded to prevent any other instrument from accidentally being attached.

The big question in the minds of hospital administrators is "why use robots in surgery?" Manufacturers and surgeons reply, "robotic arms don't have tremor so they can remain steady at all times," recalls one surgeon. "Robotic wrists make it easier for surgeons to manipulate tissue and work from all kinds of angles. They can reach around and get to places that would be harder to get to otherwise."

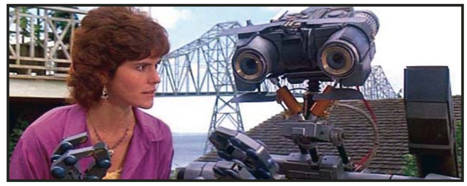


Figure 8. Short Circuit.

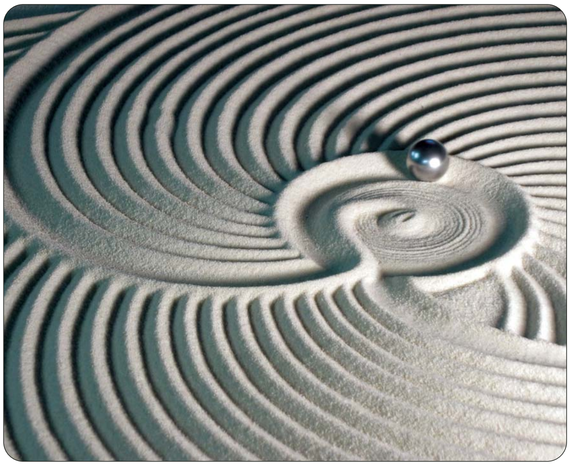
Robotic surgical systems also improve depth perception giving surgeons three-dimensional vision, compared with the two-dimensional vision they would normally get with endoscopic procedures and a standard TV monitor. So far, there haven't been indications that robotic surgery is any riskier than standard laparoscopic procedures, and there haven't been any patient injuries or deaths related to robotic system failures.

The FDA requires manufacturers to train surgeon users before they can use robotic surgical systems on patients. It has been reported that it takes 12-18 patients before surgeons feel comfortable and before surgeons are able to perform the procedures as quickly as with standard techniques.

It is likely to be several years before robotic surgery is widely available for all types of surgical procedures. Simple gall bladder removal and even more complex prostate removal procedures have been well proven with both types of systems, but it is up to the surgeons to finally accept an automated method that they have been doing manually for years. **SV**

Osiris

Lawrence Feir



Osiris is part robot, part art. The living sculpture is continually shifting the pure white sands forming mystical patterns in its four foot by four foot domain. The robot was designed and built as a prototype science museum exhibit.

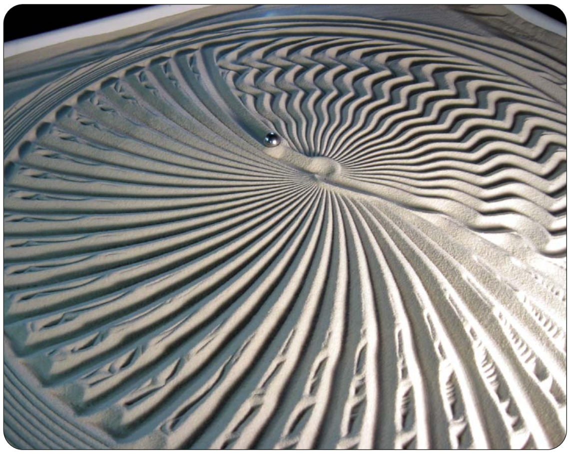
The patterns start out as computer code and contain geometric and mathematical equations. These are then "plotted" by the robot, producing some very interesting designs similar in some respects to crop circles. The Name Osiris comes from the Egyptian god of the sand. He was also known as the god of the underworld. The origin of his name is as much a mystery as these magical ripples in the sand.

Osiris is a two-axis machine working in a polar coordinate system. Power comes from a pair of stepper motors driving the axis through a precision Bodine Electric gearbox. Stepper drivers are Parker Compumotor and are commanded by a PC running Mach 2 CNC software. (**www.artofcnc.ca**)

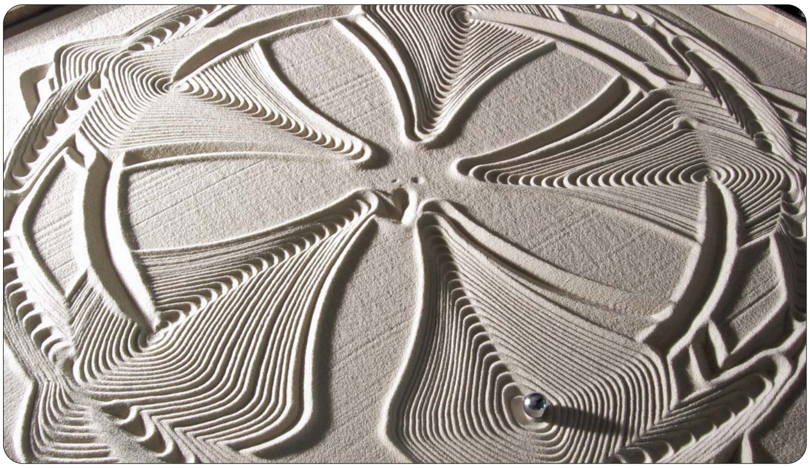
Osiris and its offspring may one day replace fountains and static sculpture in museums and public areas.

Lawrence Feir is an artist and fabricator living in Greensboro, NC. His company, Innovation Robotics, develops technological apparatus for museums, film and television.

Bodine Electric: **www.bodine-electric.com**
Parker: **www.parker.com**
Mach 2: **www.artofcnc.ca**



Lawrence Feir
Innovation Robotics
2022 Walker Avenue, Greensboro, NC 27403
336-274-4846 • www.innovationrobotics.com



Send us a high-res picture of your robot with a few descriptive sentences and we'll make you famous. Well, mostly. menagerie@servomagazine.com